The Dissertation Committee for Swati Rallapalli
certifies that this is the approved version of the following dissertation:

Mobile Localization: Approach and Applications

Committee:

Lili Qiu, Supervisor

Donald Fussell

Simon Lam

Venkat Padmanabhan

Yin Zhang

Mobile Localization: Approach and Applications

by

Swati Rallapalli, B.Tech., M.S.Comp.Sci.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2014

To Mummy, Daddy, Shreyas and Sarat who have been there and loved me no matter what.

Acknowledgments

Getting a doctorate has been a really challenging journey, where sometimes, taking every step ahead started looking hard, without the constant encouragement and cheering from my husband Shreyas. Thanks is a small thing to say to a person who has not just understood my dreams and goals but also made every single sacrifice I have made in achieving them. He has traveled to Austin so many times while working in Seattle, always cheerfully, and made sure no grad school worries or depressions got to me so that I was happy throughout.

My younger brother Sarat, and I overlapped at UT Austin. Those two years were the most fun. We went to movies, played and hung out together. After he came to the US, I never missed home. I am really glad that I got to spend time with him during my grad school. He brings the fun element into my every day routine and urges me to get into good habits that I tend to neglect — like exercising and staying fit. I hope he continues to do that and never gives up on me.

My mom R. V. Ramana and dad R. V. Prasada Rao are the people who have constantly encouraged us to always aim higher than is easy to achieve. My mom, a lecturer by profession, gave up her career to make sure me and my brother had a comfortable upbringing. She put us ahead of everything,

and the minimum we can give her in return is a reason to feel proud of the upbringing she has given us. My dad a chemical engineer by profession, is the reason I even took up engineering. As a kid, I was in awe to see him do complex calculations mentally. I thought that if I became an engineer I would be as sharp as him. It took me a long time to understand that becoming as sharp as him was not just about becoming an engineer. His role does not end at merely inspiring me to be an engineer. He spent several hours with me actually motivating me to study and work hard. Living in Mumbai, despite traveling for three hours each day to work, he still somehow made time to read through our books, understand the problems we were solving and help us whenever we needed guidance.

I do not have enough words to thank my adviser Lili without whom this dissertation would not be possible. *"Gurur Brahma Gurur Vishnuhu Guru Devo Maheswaraha Guru Saakshaat Parabhrahma Tasmai Sri Gurave Namaha"*. This Sanskrit verse we were taught in India says that: "A teacher is a representative of God as he creates, sustains knowledge and destroys the weeds of ignorance in a student, I salute to such a teacher." Today, I appreciate its true meaning as Lili through the past six years has shaped my career and thus my entire life. After my mother and father, she is probably the person who has had the most impact in molding my thinking and character. When I first came to UT Austin as a Masters student, I was not sure of enrolling for a Ph.D degree. In my first semester here, I took her graduate class in Wireless Networking. It is through her questions in class and the way

she lead us to answers that built the curiosity and confidence in me, to take up research towards my Ph.D. Ever since then, she has been motivating me through example. Most of what I know about research, I have learned from her, and among all the other things I have learned from her about research, I have learned to work hard and to not give up easily. She is a perfectionist, and from her, I have learned to be thorough about everything I do. She has always encouraged me to think about problems in multiple areas even outside of my dissertation topic, and now I realize how important it is, as it is a critical skill in research, to be able to pick up a new area quickly and be able to apply our knowledge in one area to solve problems in another. She was always available to us whether to chat or discuss or brainstorm, she never let us feel that she was too busy to be approached and I think that is phenomenal. If someday I could be as professional as her, I would consider it an achievement. I want to thank her for being patient with all my mistakes right from the beginning and yet guiding and showing me the right path through out.

I was also fortunate to get Yin's guidance on many of my projects. Meetings with him have been enriching. I got to learn a lot from his intuition and experience. His sense of humor brought life into our project meetings.

I would like to thank my committee members for being accommodating and supportive throughout. Working on indoor localization, I was fortunate enough to get an opportunity to intern at Microsoft Research India under the mentor-ship of Venkat. After reading, citing and seeing the reference of his work (RADAR the pioneering work in this area), in every localization paper

I read; to learn about his vision in this area of research while working with him was truly a dream come true for me. Prof. Simon Lam has always given me advice on research, and job options after grad school. To hear about how the networking field evolved over the years and the associated tiny stories, through his classes I took has been really enjoyable. Prof. Donald Fussell has always asked me great questions in such a manner that discussions became a lot more fun. A lot of the tools and techniques we use in analyzing data for localization, I have learned from his graduate class on Computer Graphics. I will remain indebted to my committee members for inspiring, motivating and guiding me throughout.

Through my grad school I have had the privilege to work with and learn from my wonderful mentors during internships. I got the opportunity to work closely with Krishna at Microsoft Research India, from whom I have learned so much about localization and the relevant areas — the issues with practical deployments and the motivation behind current solutions. His energy and passion for research really motivate me. He is always willing to discuss and help. He thinks out-of-the-box and encourages me to do so. The clarity he has in analyzing what novel aspects a project could bring to the table is amazing.

My mentors at Narus — Mario and Han, have spent several hours brainstorming and discussing ideas with me. I thank them for always being available to answer my questions and to guide me. My mentor K. K. at AT&T Research was critical about our presentations and coaxed us to present our

ideas clearly and this really helped me come out of my shell and talk, which I have realized over years is a very important skill to acquire in research. Rittwik and Leo at AT&T Research were always helpful in getting us the resources for our research. Thanks to my internship at AT&T Research I developed an interest for and learned about a completely new area — auction design.

Through the course of my Ph.D. I had the chance to work with my superb student co-authors. Yi-Chao simply amazes me by working hard, through deadlines without sleep and always with a smile on his face. Wei, always encouraged me to think harder by asking great questions, discussions with him have been great fun. Working with Aishwarya, on Physical Analytics project has been a pleasure. I want to thank Gene, Apurv, Eric, Mikie, Sangki, Owais, Mubashir, Vacha, Srinath, Hongkun and all my colleagues and friends at LASR for their inputs on my research, for helping me while working on things they were familiar with and for their support, motivation and company throughout.

Life in the CS department was made easy for me as a student by our incredible administrative team — Lydia, Katherine, Leah, Sara, Lindy, Phyllis, who have taken care of all the difficult stuff and let me focus on academics alone.

I have to thank my support system in Austin for making the city so much fun for me. Archana and Mahesh first helped me settle down in Austin and knowing them has been a pleasure. Yesha and Neeraj have been the reason for so many fun weekends we spent together. All the good food, picnics, playing Hindi songs, random trivia, random chats and celebrating festivals together

— they are like family in Austin, thank you both! Lavanya and I have had several meaningful as well as not so meaningful chats, she has given me the sisterly affection, while always being around and caring. Shruthi and Rutvi are wonderful room-mates I had during the first two years at Austin. I will not forget how I came home from lab at 2 am one night before a MobiCom deadline to find my dinner served on the table, covered, waiting for me by my room-mate Rutvi. Thank you and I hope and pray that these friendships last for life!

The best things in life come after the worst of struggles and grad school has been a long journey with its share of fun, hard work, disappointments, learning, collaborations, discussions and finally the successes. I would like to thank absolutely everyone who has helped make this struggle easier for me.

Mobile Localization: Approach and Applications

Swati Rallapalli, Ph.D.
The University of Texas at Austin, 2014

Supervisor: Lili Qiu

Localization is critical to a number of wireless network applications. In many situations GPS is not suitable. This dissertation (i) develops novel localization schemes for wireless networks by explicitly incorporating mobility information and (ii) applies localization to physical analytics i.e., understanding shoppers' behavior within retail spaces by leveraging inertial sensors, Wi-Fi and vision enabled by smart glasses.

More specifically, we first focus on multi-hop mobile networks, analyze real mobility traces and observe that they exhibit temporal stability and low-rank structure. Motivated by these observations, we develop novel localization algorithms to effectively capture and also adapt to different degrees of these properties. Using extensive simulations and testbed experiments, we demonstrate the accuracy and robustness of our new schemes.

Second, we focus on localizing a single mobile node, which may not be connected with multiple nodes (e.g., without network connectivity or only

connected with an access point). We propose trajectory-based localization using Wi-Fi or magnetic field measurements. We show that these measurements have the potential to uniquely identify a trajectory. We then develop a novel approach that leverages multi-level wavelet coefficients to first identify the trajectory and then localize to a point on the trajectory. We show that this approach is highly accurate and power efficient using indoor and outdoor experiments.

Finally, localization is a critical step in enabling a lot of applications — an important one is physical analytics. Physical analytics has the potential to provide deep-insight into shoppers' interests and activities and therefore better advertisements, recommendations and a better shopping experience. To enable physical analytics, we build ThirdEye system which first achieves zero-effort localization by leveraging emergent devices like the Google-Glass to build AutoLayout that fuses video, Wi-Fi, and inertial sensor data, to simultaneously localize the shoppers while also constructing and updating the product layout in a virtual coordinate space. Further, ThirdEye comprises of a range of schemes that use a combination of vision and inertial sensing to study mobile users' behavior while shopping, namely: walking, dwelling, gazing and reaching-out. We show the effectiveness of ThirdEye through an evaluation in two large retail stores in the United States.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Location information is critical to a number of wireless network applications; from sensor network applications like environment surveillance and habitat monitoring to smart-phone applications like location based reminders, or in-store shopping assistants.

Although GPS is widely used for location determination, in many scenarios GPS is not suitable either due to energy constraints or the lack of line of sight to the GPS satellites. Localization in the absence of GPS is a challenging problem as we try to leverage signals like Wi-Fi and have to deal with incomplete information, noisy measurements, and variations in signal propagation based on the environment.

This dissertation advances the state-of-the-art in the area of localization in multi-hop networks as well as single wireless nodes (which includes single-hop wireless networks such as infrastructure wireless networks or a single node, which is not even connected to a network), by leveraging mobility and vision to extract additional information to address the above challenges. Further, it applies localization to understand shopper behavior within retail

spaces to provide a better shopping experience through product guides, recommendations and shopping list reminders.

1.2 Motivation

Location information is critical to a number of sensor network applications like environment surveillance [7, 97, 103], habitat monitoring [21, 49], military reconnaissance [41], underwater surveillance [13, 100] and a number of smart-phone applications like location based reminders, guides to product locations within retail stores, or friend discovery in public places like airports. Localization, hence, is a fundamental operation in wireless networks.

The Global Positioning System (GPS) [38] is widely used to obtain location information. But GPS does not work in many environments due to the lack of line of sight to the satellites (*e.g.*, indoors, under the ground, or in a downtown canyon). Moreover, it is often too expensive to equip every wireless device (*e.g.*, low power sensors) with a GPS receiver.

Localization in Mobile Networks: The limitations of GPS have motivated researchers to develop many localization schemes to infer locations based on cheap hardware and wireless measurements (*e.g.*, [11, 42, 44, 60, 71, 92, 93, 107]). For these schemes, localization accuracy depends heavily on the amount of information that one can extract to constrain the set of possible locations of a given node.

A lot of research has gone into developing localization schemes for static

networks [92, 93, 11, 71, 77]. For example in the case of static networks, there are many interesting proposals on deriving location constraints from signal strength [11], angle of arrival [71], and difference in arrival time of different types of signals [77]. Schemes that consider inter-node relationships [92, 93] have also been proposed for localization in a static multi-hop wireless network. However, wireless networks are fundamentally mobile. To localize mobile networks, one possible approach is to treat the nodes during different time intervals independently and apply existing localization schemes at each interval; but such an approach is unlikely to achieve high accuracy because it does not fully capture the relationship between a particular node's locations at different snapshots of time. In this dissertation we explicitly build techniques for localization in mobile networks.

Location based Physical Analytics: Once we have location information, a number of applications become feasible. One of the important applications is *physical analytics*. Physical analytics involves tracking mobile users' interests in the physical world (much like web analytics with foot-prints taking the place of click-streams) and has the potential to provide deep-insights into users' interests and activities to help provide better advertisements and recommendations.

Web analytics is a multi-billion dollar industry today, providing tools that measure and analyze users' online browsing behavior. A typical tool will track webpage-related events, such as page views, clicks, and the degree to which different landing pages are associated with online purchases. Most

purchases today, however, occur in the physical realm, *e.g.*, at retail stores and shopping malls. While many retailers track shoppers' purchases (*e.g.*, as part of customer loyalty programs) in order to perform analytics, these systems do not provide insight into shoppers' behavior *during* the shopping process, which we refer to as *physical browsing*. For example, which sections of the store did the shopper spend most of his or her time in? Which products did the shopper express interest in by gazing at them or reaching out for them? How many shoppers reached out for competing products A and B, say to compare these?

We believe that access to physical browsing information of shoppers in retail stores can not only provide crucial insights into shoppers' needs and interests but also reveal the effectiveness of the store layout itself. Furthermore, such *physical analytics* that track in-store shopping behavior could also be combined with online information such as web browsing history or a shopping list to generate automatic alerts, say when there is a match between the physical browsing context and the online information.

1.3 Challenges

The general challenges faced are discussed in this section. Localization is critical for physical analytics in order to understand where the user is walking, stopping and spending time. Thus most challenges are common to localization and physical analytics, and we discuss them together in this section.

First, a majority of localization algorithms rely on Wi-Fi measurements,

i.e., received signal strength (RSS) or channel state index (CSI) (i.e., more fine-grained measurements across the OFDM subcarriers – unlike RSS which is one measurement per Transmit-Receive antenna pair), measurements for localization. Mobility-induced fading can make RSS or CSI measurements and in turn the derived distance measurements much less accurate. Further, if the measurements exhibit a large variation, it is infeasible to capture the entire distribution of the measurements at a given location while the nodes are moving. At several locations on the trajectory we may record no measurements at all.

Second, in the case of fingerprinting based schemes for localizing single wireless nodes, measurements taken from a single point exhibit considerable aliasing. For example, existing Wi-Fi based localization schemes use RSS or CSI measurements from nearby access points (APs) to determine locations. But it is common to have multiple locations that see similar RSS or CSI.

Third, in multi-hop networks the anchor nodes (*i.e.*, nodes with known co-ordinates like wireless Access Points) maybe multiple hops away from the mobile node to be localized. Thus any errors in distance measurements cumulatively add up, rendering localization even less accurate.

Fourth, different localization algorithms assume different settings. For example, RADAR [11], Horus [117] assume prior profiling of the environment, ArrayTrack [111] assumes modified Access Points (APs), Zee [79] assumes map information. Based on the application scenario, such information may not be available. For example, in our ThirdEye system for physical analytics, we

assume no infrastructure support, and no prior profiling to be able to cover a wide foot-print and track the user while he visits different retail spaces.

Fifth, while dealing with product locations in retail stores for physical analytics, it is challenging to keep them up-to-date as product locations could frequently change within a store, for e.g, stores cater to different shopping priorities during different occasions like Halloween versus Christmas.

Finally, mobile devices are resource constrained. Thus energy efficiency is an important consideration, for designing the data collection and the processing components for localization and physical analytics.

1.4 Approach

Localization in Multi-Hop Mobile Networks: First, we consider a multi-hop mobile network (e.g. a mobile sensor network deployment) where some nodes have known locations referred to as *anchor* nodes and the remaining nodes need to be localized. There are only a limited number of recent works that explicitly consider the problem of localization in multi-hop mobile networks (*e.g.*, [10, 44, 85, 98]). However, these works do not fully exploit the location information available in each time interval. For instance, existing localization schemes for multi-hop mobile networks only use the maximum speed to constrain the distance between a node's positions during two consecutive intervals. Moreover, their evaluation uses simulation based on synthetic mobility traces. It is not clear how well they perform under a real mobility pattern

or in a real network.

In this work, we develop novel techniques to accurately localize nodes in multi-hop mobile networks by exploiting structural properties of the underlying node mobility patterns. Intuitively, the nodes' coordinates over time are not independent but have certain relationships. By exploiting such relationships, we derive additional constraints to further narrow down the set of feasible locations and thus significantly improve localization accuracy.

To extract the relationships between nodes' coordinates across different time intervals, we first analyze the characteristics of a number of real mobility traces, including (i) taxi and bus traces in San Francisco [18], Shanghai [121], and Seattle [89], (ii) human mobility traces [45], and (iii) ZebraNet traces [118]. Interestingly, while these traces capture rather different mobility scenarios, they all exhibit two pronounced structural properties: (1) *low-rank structure*, *i.e.*, the matrix formed by node coordinates over time can be well approximated by a low-rank matrix; and (2) *temporal stability*, *i.e.*, the direction and speed of the same mobile node is often similar at adjacent intervals. Our results suggest that these structural properties are also present in synthetic traces generated using two commonly used mobility models: the standard random waypoint model [16] and the modified random waypoint model [44].

Motivated by these observations, we develop a general framework that simultaneously incorporates location constraints during each interval and exploits the low-rank structure and temporal stability in mobility. Specifically, we formulate the localization problem as an optimization problem that mini-

mizes (i) the fitting error between measured distance and estimated distance based on node coordinates, (ii) the error in approximating the estimated coordinate matrix using a low-rank matrix, and (iii) the changes in velocity between nodes in adjacent time intervals. Essentially, (i) leverages the location information during each time interval of the mobile network, which can be obtained using existing localization techniques for static networks (*e.g.*, [11, 71, 77, 93]), (ii) reflects the low-rank nature of the coordinate matrix in a mobile network, and (iii) captures the temporal stability in node movement.

Using this framework, we develop three novel localization schemes for mobile networks: (i) Low Rank based Localization (LRL), which exploits the low-rank structure in mobility, (ii) Temporal Stability based Localization (TSL), which leverages the temporal stability, and (iii) Temporal Stability and Low Rank based Localization (TSLRL), which simultaneously takes into account the temporal stability and the low-rank structure of mobility. These localization schemes use distance estimation between neighbors and we call them *range-based* localization schemes. In addition, based on the general framework, we also develop *range-free* variants of these schemes, namely, LRL-RF, TSL-RF, TSLRL-RF. These variants use only network connectivity for localization and can therefore support nodes that are unable to obtain accurate distance estimation (*e.g.*, due to lack of RSS measurements).

We extensively evaluate our localization schemes using (i) simulations based on both synthetic mobility traces and several real mobility traces, and (ii) experiments in a sensor network testbed consisting of 25 or 36 *mica*2

motes. The simulation results show that our new schemes significantly outperform the existing schemes: they consistently improve accuracy by 50%-90% over Centroid [17], 10%-80% over MDS [93], 30%-90% over Sextant [39], 30%-90% over MCL [44], and 20%-80% over MSL* [85], where Centroid, MDS, and Sextant are well-known localization schemes for static networks, MCL and MSL* are state-of-the-art localization schemes for mobile networks. The performance improvements of our schemes in the testbed experiments are 60%-75% over Centroid, 60%-75% over Sextant, 40%-55% over MDS, 65%-80% over MCL, and 50%-70% over MSL*. Moreover, our schemes are highly robust and can achieve high localization accuracy even in the presence of measurement noise, irregular topologies, and different mobility patterns.

Localization of single mobile nodes: Next, we consider localization of single mobile nodes. This includes single hop wireless networks such as infrastructure wireless networks or a single node, which is not even connected to a network. For example a smart-phone, that needs to be localized within areas where GPS maybe unavailable or only intermittently available. Fingerprinting based localization schemes are commonly proposed for these scenarios [11, 117] and use fingerprints from a single point for localization. While it is a natural way to localize nodes based on point based signatures, they face aliasing issues as described in Section 1.3. Thus, we explore the potential of using measurements from trajectories as location fingerprints. Specifically, we let mobile devices collect measurements while moving with the users, and use the measurement time-series as a signature for localization. The measurement can

be geo-magnetic field, received signal strength (RSS), or channel state information (CSI). Localization is then performed by (i) collecting training traces which associate measured magnetic field, RSS, or CSI with locations (*e.g.*, GPS coordinates), (ii) matching a new measurement trace without location information with one of the training traces (*i.e.*, trajectory matching) to identify the right trajectory, and (iii) narrowing down to the exact location on the chosen trajectory, based on the matched point on the training trace (*i.e.*, localization). Since measurements collected over a trajectory offer more information, trajectory matching is likely to be more accurate than point matching; moreover, localizing a point on a trajectory is much easier than localizing a point in an area. Therefore, by leveraging more information and decomposing the localization problem into the two easier tasks, we improve accuracy.

While it is evident that trajectory based localization requires more information than point based localization, it is not necessarily more expensive to collect such data. Since people walk and drive in a few common trajectories (*e.g.*, corridors in office building, aisles in grocery stores, and lanes on roads), we can populate our training database by collecting measurements from users' natural walking and driving patterns.

This approach, however, is faced with several significant challenges. First, it is hard to exactly match testing and training traces even if they are collected from the same trajectory, due to different speeds, similar but not identical trajectories, different devices, environmental changes, and different levels of noise. Second, localizing a point on the trajectory requires a highly

accurate alignment between the testing and training traces, which is also challenging due to the differences in the testing and training data. Even a small amount of noise could potentially lead to incorrect alignments.

The core requirement of above steps (ii) and (iii) is an accurate and robust algorithm to match two time-series that may be noisy and subject to various perturbations arising from different mobility, devices, and environmental factors. To find an alignment between the two time-series, we apply the well known Dynamic Time Warping (DTW) [69] algorithm, widely used in signal processing, speech recognition, and signature recognition [61, 86, 78]. To improve robustness against different speeds and noise, we develop a few enhancements: (i) we perform multi-level wavelet decomposition on the measurement data and search for the wavelet level that minimizes the distance between the two time-series, (ii) we cluster the training traces from the same trajectory to capture different patterns that a trajectory may exhibit, (iii) we enable continuous localization by combining trajectory matching and a novel procedure to detect the end point of a training segment, (iv) we use a number of techniques to address the singularity problem common in traditional DTW, and use incremental DTW to reduce the computation cost.

Location based Physical Analytics: Finally we focus on leveraging location information for physical analytics.

One approach to understanding user shopping behavior and interests within indoor retail spaces, would be for the retail stores themselves to leverage this infrastructure like the in-store surveillance videos to mine such informa-

tion. However, many retail stores today neither possess the expertise nor the resources to gather and analyze tons of surveillance data for this purpose. Furthermore, this approach would *not* allow aggregating data across a user's online and physical browsing or indeed even just physical browsing across different stores that a user may visit since an individual store is unlikely to be willing to share information with potential competitors. On the other hand, the advent of smart glasses such as Google Glass [37], equipped with a camera, inertial sensors and Wi-Fi, enables user-driven data gathering, both (privately) tracking an individual user's visits across stores and crowd-sourcing store layout information based on the visits by a community of users. Furthermore, for an individual user, there would be an incentive to participate since their physical browsing information could be used directly for their own benefit (*e.g.*, for receiving personalized alerts from a digital personal assistant) rather than being shared for the stores' business purposes. For it to scale to a wide footprint, such a user-driven approach should ideally *not* depend on any support from the disparate stores that the user may visit.

In this work, we present a system called ThirdEye to address the above issues. ThirdEye only uses image, inertial sensor, and Wi-Fi data crowdsourced from shoppers wearing smart glasses to track the physical browsing of shoppers, without requiring any input from either the store owners or from the shoppers themselves. ThirdEye includes three main components: product layout inferencing, shopping behavior classification and user attention identification. We discuss them briefly in turn.

Layout Inferencing: Inferring the layout of the products becomes an important goal for ThirdEye to achieve our goal of tracking physical browsing. For instance, an application might wish to generate a proximity-based alert for an item that is quite close yet out of view (*e.g.*, it might be right behind the shopper). Moreover, there would be no vision-based sensing for shoppers who only have a smartphone and no Glass.

Since we do not rely on the store to provide us with a map, we design a novel scheme called *AutoLayout* that fuses video, Wi-Fi, and inertial sensor data from the Glass devices worn by various shoppers, to simultaneously track them while also constructing and updating the *product layout* in a virtual coordinate space (Section 5.1). The advantage of this scheme is that we can localize not only Glass-enabled shoppers but also shoppers who carry only smartphones using Wi-Fi and inertial sensor data, albeit with reduced accuracy.

Shopping behavior classification: By analyzing over 50 video streams from 7 shoppers wearing the Google Glass, shadowing them as they were shopping and interviewing them, we have identified four distinct behaviors that shoppers exhibit at retail stores: *purposeful walking*, *dwelling*, *gazing*, and *reaching out*.

A shopper who knows what they wish to purchase or is in a hurry would likely walk purposefully to the relevant section. They might then dwell in a specific area (*e.g.*, an aisle) for a length of time. Such dwelling is characterized by slow, unpurposeful movement, say with the shopper ambling around nearby locations. Thereafter, having further narrowed down the products of interest,

the shopper might gaze for a little time, looking steadily at a product or a set of closely-placed products, perhaps to search for or make up their mind about the products. Finally, the shopper might reach out for a product of interest with his or her hands, either to examine it more closely or to purchase it.

ThirdEye uses video and inertial sensing from smart glasses to determine whether the shopper is walking, dwelling, gazing, or reaching out. A key challenge is in tracking all of these activities accurately and in an energy-efficient manner. Since continuous video is extremely expensive in terms of energy consumption, our general strategy is to rely on inertial sensors as much as possible and to trigger vision only when necessary. These behaviors are explained in greater detail in Section 5.3.

User Attention Identification: Once we establish that the shopper is gazing, the video feed can be used to identify the products within the shopper's field of view using existing techniques such as Google reverse image search. However, as we discuss in Section 5.4.2, our experiments reveal the presence of as many as 16 products in the shopper's field of view. An important task then is to accurately identify the shopper's item of focus among the many products within view, *i.e.*, which item the shopper is looking at.

A naive approach would be to use the distance between an item and the center of the field of view as an (inverse) measure of the likelihood of it being the object of the shopper's focus. However, as we discuss in Section 5.4.2, the item of interest may not lie at the center of the field of view, depending on the orientation of the shopper's head. By analyzing ground truth data from

several real shoppers, we develop a model for the likelihood of an item being at the shopper's focus, depending both on its position within the field of view and the orientation of the shopper's head, obtained from the Glass.

1.5 Summary of Contributions

In summary, here are the following major contributions of the dissertation:

- This dissertation advances the state-of-the-art in localization algorithms for mobile networks, by explicitly leveraging mobility to extract additional information in two different scenarios: (i) multi-hop mobile networks and (ii) single mobile nodes.
- Develops a system ThirdEye, that enables physical analytics by inferring shopper locations and product locations in a zero-effort manner by leveraging shoppers wearing smart glasses. Moreover, it recognizes and defines interesting shopping behaviors, namely, *walk*, *dwell*, *gaze* and *reach-out* and develops algorithms to automatically identify these behaviors to help infer the interests of the shoppers.

1.6 Dissertation Outline

Chapter 2 overviews the related work. Chapter 3 details our approach on leveraging the structure in mobility for localization in multi-hop networks, Chapter 4 describes localization of single mobile nodes using trajectory fingerprints, Chapter 5 details our work on localization based physical analytics

and finally Chapter 6 presents the conclusions.

Chapter 2

Related Work

This chapter presents the related work. Section 2.1 talks about localization in multi-hop networks, Section 2.2 presents the work related to localization of single wireless nodes and finally Section 2.3 describes the work in Physical Analytics related areas.

2.1 Localization in Multi-Hop Networks

The works we review below explicitly localize nodes in a multi-hop network leveraging the multi-hop constraints. The related work in this section is further classified into (i) localization in multi-hop static networks, (ii) localization in multi-hop mobile networks and (iii) compressive sensing.

2.1.1 Multi-Hop Static Networks

Most works on localization in multi-hop networks target static wireless networks. For example, the authors in [87] develop a distributed localization approach that iterates through two phases: ranging and estimation. During the ranging phase, each node estimates its distance to its neighbors; and during the estimation phase, it estimates its location based on the ranging

information and the locations of its neighbors whose positions have been determined. To limit error accumulation in [87], the authors in [88] enhance the previous approach by formulating the problem as a global non-linear optimization problem. Shang *et al.* [93] apply multi-dimensional scaling (MDS) to determine location in a centralized fashion. MDS estimates the distance matrix using shortest path between two nodes and then finds the nodes' coordinates by solving a least square problem: $min_{x_1,\ldots,x_n} \sum_{i<j}(\|x_i - x_j\| - D_{i,j})^2$, where $\|x_i - x_j\|$ is the estimated distance based on their coordinates and $D_{i,j}$ is the measured distance. MDS performs poorly when the shortest path distance does not correlate well with the Euclidean distance, which is common in irregularly shaped networks. In [68], the authors propose robust quadrilateral for localization. It finds sets of four nodes that are fully connected, and localizes the fourth node based on the positions of the other three nodes. Robust quadrilateral conditions have to be satisfied by the fourth node in order to prevent error accumulation. Therefore it improves accuracy at the cost of leaving some nodes un-localized. Biswas *et al.* [15] formulate the localization problem as a semidefinite program, and later develop global optimization approaches [14]. Wang *et al.* [106] further propose a low-rank semi-definite programming (SDPLR) formulation of the localization problem.

Unlike most of the previous approaches, which represent inferred locations using points, Sextant [39] denotes inferred locations as regions that satisfy distance measurements. In particular, when node i hears from node j, it extracts a positive constraint that their distance is within the communica-

tion range. When node i does not hear from node j, it extracts a negative constraint that their distance is larger than the communication range. A node then estimates its location by finding a region that satisfies both positive and negative constraints. All points in a region are considered equally likely locations for a node. [104] proposes probabilistic region-based localization that uses a dynamic mesh to represent a region and computes the probability for a node to reside in different parts of the region. This improves accuracy over Sextant at the cost of higher computation time. All these approaches are designed for static wireless networks.

2.1.2 Multi-Hop Mobile Networks

Compared to the significant related work on localization in static multi-hop networks, there are considerably fewer works on localization in multi-hop mobile networks. Among the few existing works, [44] is the first localization scheme for multi-hop mobile networks. It uses a sequential Monte Carlo Localization (MCL) method to localize mobile sensors. A node uses its previous location and maximum speed to generate possible current coordinates. Then it filters out infeasible locations using the current connectivity information. Since MCL only extracts information for nodes that are either direct neighbors or 2-hop neighbors from anchor nodes, it requires a high anchor density to work well. Moreover, its sampling technique converges very slowly as reported in [85] and observed in our own evaluation. [10, 85, 98] propose several enhancements over MCL sampling. Among them, MSL* [85] performs the

best. MCL only supports mobile networks, and MSL* [85] modifies the sampling procedure to support both static and mobile networks. Moreover, it lets nodes use information only from the neighbors that have more accurate coordinates to speed up convergence and improve accuracy. Nevertheless none of these schemes fully leverage the temporal or spatial information available in the network.

2.1.3 Compressive Sensing

The localization problem in multi-hop networks is related to the general area of compressive sensing in that both aim to recover the unknowns based on partial observations. To cope with the under-determined nature of the problems, many algorithms have been proposed in the area of compressive sensing. Their effectiveness depends on their ability to exploit the unique structure in the data. For example, [30, 31, 73] exploit sparsity, [20, 32] exploit low rank structure, and [120] exploits spatio-temporal stability. Motivated by these, our work exploits the low rank and temporal stability of coordinate matrices for mobile network localization for the first time.

2.2 Localization of Single Wireless Nodes

When the wireless nodes to be localized do not communicate with each other, multi-hop information may not be available. For example, users' smart-phones in a shopping mall will easily be able to measure signal strengths from the deployed access points, but not from other smart-phones in the environ-

ment. The related work in localizing single wireless nodes is discussed by further dividing them into localization of single (i) static nodes and (ii) mobile nodes.

2.2.1 Single Static Nodes

Large amount of localization work has focused on static single nodes. They use point-based measurements for localization. They differ in the types of measurements and analyses. For example, RADAR [11] uses triangulation based on signal strength measurement. Cricket [77] uses the difference between the arrival time of radio and ultrasound signals to estimate distance. VORBA [71] determines location based on the angle of arrival measurements from 802.11 base stations. [17] proposes Centroid, which estimates the location of a node as the center of all neighboring anchor nodes. When there are no anchor nodes nearby, a node estimates its location as the center of the area. Horus [117] exploits the stochastic nature of the RSS map and applies a maximum likelihood based scheme. [48] uses CSI for location distinction (*i.e.*, determine if a node has moved from its position). PinLoc [91] uses CSI signatures from specific spots for localization. All these schemes require significant training data in order to address aliasing. EZ [22] minimizes profiling effort by using measurements from unknown locations along with a few from known locations to extract propagation constraints and applies a genetic algorithm to solve them. CUPID [90] leverages the direct path signal to estimate the distance and the angle of arrival to reduce inaccuracies from multipath effect.

All of the above schemes can be extended for the case of single mobile nodes, by treating each snapshot as independent and localizing the nodes at each snapshot using the above schemes. But this approach ignores the constraints between the location of nodes across time and thus is unlikely to yield very high accuracy.

2.2.2 Single Mobile Nodes

There exist some works that use trajectory information for localization of single mobile nodes. For example, [105] uses sensor information (*e.g.*, accelerometer) to first locate landmarks and then localize new points by inferring the number of steps from the landmarks. [112] tracks a mobile device-free user based on how they disturb the radio signal patterns. It is unclear how the scheme will work in crowded environments, where we do not know which user caused the observed changes. Walkie-Markie [94] builds a crowd-sourcing system to construct indoor pathway maps by using landmarks on a trajectory that record the strongest signal strength from an AP. The robotics community uses Kalman Filters for Simultaneous Localization and Mapping (SLAM) (*e.g.*, [29, 95]). It assumes that location follows a Gaussian distribution, which does not hold in practice. Zee [79] uses a particle filter based approach for localization. It eliminates improbable routes based on the map. Kalman filter, HMM, and particle filter based approaches all use the current trajectory, while we use both the current and previous trajectories, thus leveraging more information for accurate localization. LiFS [113] removes the necessity of manual profiling

by creating a fingerprint space where the fingerprints are distributed based on their pairwise physical distance estimated from real user mobility. LiFS, like our work also leverages past trajectory information, but only to count steps between fingerprints that are collected at static points. Our fingerprints on the other hand are collected continuously as the user is moving.

[105] and [101] are schemes that leverage current trajectory for localization as explained above, and in addition, use magnetic field to identify landmarks. Since magnetic field is only leveraged from a few points they can face aliasing issues. Aliasing is especially significant for magnetic field since it is common for multiple locations to have similar magnetic field. [47] is a start-up that uses indoor magnetic field anomalies for indoor localization. Their exact technology is not disclosed to the public. [99] uses the magnetic field for indoor trajectory matching. Our work advances state-of-the-art in the following ways: (i) performing point localization (beyond just trajectory matching), (ii) enhancing efficiency and accuracy using multi-level wavelet analysis, (iii) applying our approach to both Wi-Fi and magnetic field and showing its benefit both indoors and outdoors.

2.3 Physical Analytics

Physical analytics is a broad area, encompassing several components, like localization, vision-based sensing, robotics, human activity sensing, and physical analytics in retail (including the work in start-ups).

2.3.1 Indoor Localization and Sensing

There has been extensive work in indoor localization and sensing and these works have been discussed earlier in this chapter. Here, we just touch upon the differences of our localization and product layout inferencing scheme (AutoLayout) in ThirdEye system from the existing works. Despite much work on Wi-Fi localization, existing work can achieve high accuracy only at the expense of a high calibration cost or requiring additional information or modifications to Wi-Fi Access Points (APs) (*e.g.*, Horus [116] requires detailed profiling of the indoor space ahead of time, Zee [79] requires maps and ArrayTrack [111] requires modified APs). In comparison, our system achieves semantically meaningful information, such as item or category labels without requiring external input (*e.g.*, a map) or other human input (*e.g.*, manually-assigned labels from crowd-sourcing).

SurroundSense [9] proposes an interesting approach that fingerprints locations based on visual, acoustic, and motion signals. The goal is to determine which store the user is located in. In comparison, ThirdEye seeks to identify the user's context in a more fine-grained manner than just the identity of the store that they are in (*e.g.*, whether the user is gazing and if so at which item) and do so without requiring any fingerprinting in advance.

CrowdInside [8] presents a method for constructing indoor floor plans by leveraging the movement of users carrying smartphones. It uses dead-reckoning together with anchor points to prevent error accumulation. The anchor points are defined by the unique inertial sensor signatures corresponding

to elevators, escalators, and stairs. However, such inertial-based anchor points might be few in number or even non-existent within a store. In contrast, by leveraging vision, ThirdEye is able to identify more accurate, item-level landmarks. Furthermore, by combining this with both inertial tracking and Wi-Fi information, ThirdEye provides a unified framework that caters to both smart glass users and smartphone-only users.

2.3.2 Vision

There is a vast body of research on vision-based place recognition and pose estimation. For example, [6] constructs a 3D model of a city based on a large number of photos collected online, which can then be used to estimate the pose (*i.e.*, position and orientation) of an observer with a given view. [55] determines the location where a photograph was taken by matching it with photographs of popular landmarks. [74] develops an approach that retrieves similar images from a visual database on a mobile device with a small memory footprint.

Vision-based approaches are generally expensive, especially if it involves building a 3D model, hence these have generally only been applied to well-known landmarks. The insides of stores are typically lacking in such distinctive landmarks and are often crowded with people, posing challenges. In ThirdEye, we make limited use of image information to recognize items rather than scenes, thereby sidestepping these challenges.

2.3.3 Robotics

There is a vast body of work on robot navigation, specifically on the classical problem of simultaneous localization and mapping (SLAM) [54]. While the robots typically have access to precise wheel odometry, ThirdEye uses only human walking and does not use specialized sensors, such as laser range-finders.

There have also been prior SLAM-like efforts in the context of human motion. FootSLAM [83] uses foot-mounted inertial sensors to track a user's walk, while FeetSLAM [84] extends this to combine the trajectories of multiple users, assuming each walk is sufficiently long. Neither uses Wi-Fi or visual information. WiFiSLAM [28] only uses Wi-Fi information, not inertial sensing. Only visual information is used in MonoSLAM [27]. To our knowledge, ThirdEye is the first system that combines inertial and Wi-Fi sensing with visual information to construct a map from walks by multiple humans, even when the individual walks are short.

2.3.4 Human-Activity Sensing

There has been much work on fine-grained human-activity sensing using wearable devices such as pedometers, heart-rate monitors, microphones, etc. [19, 34, 23]. ThirdEye focuses on a small subset of activities of relevance in the physical browsing context and enables these using smart glasses. As more wearable devices become prevalent, we could leverage those too; regardless, ThirdEye's ability to combine inertial, Wi-Fi, and visual information would

help provide physical context to the sensed information (*e.g.*, associating a jump in the user's heart rate with them gazing at a particular object on display).

2.3.5 Shopping Behavior

[50] places sensors on people to track their trajectories, and uses a clustering scheme to predict users' future behaviors (*e.g.*, fast walking, idle walking, or stopping). [53] studies 701 hours of sensor data collected from 195 in-situ customers to understand customer behavior in shopping malls. [115] monitors customers' shopping time. In comparison to these specific studies, ThirdEye provides a broad and systematic framework for combining multiple sensing modalities to track physical browsing by users.

2.3.6 Retail Analytics Start-ups

Nearbuy Systems [70] requires retail stores to deploy a customized Wi-Fi localization infrastructure to analyze in-store customers (dwell times, visit frequency, window conversions, etc.) Euclid Analytics [35] leverages existing in-store Wi-Fi infrastructure to provide similar analytics to retail stores. Neither approach provides fine-grained, item-level information. Apple's iBeacon [46] transmits location-specific messages in a store to nearby smartphones via Bluetooth Low Energy (BLE). Mondelez [96] requires retail stores to deploy cameras in shelves that use facial-recognition to provide insights into demographics of customers that browse a given product. In comparison to

these systems, which require extensive infrastructure roll-out or support and hence are meant to be deployed by store owners, ThirdEye does not require any special infrastructure and hence is amenable to crowd-sourcing. Moreover, ThirdEye provides more fine-grained analysis including user behaviors.

Chapter 3

Localization in Multi-Hop Mobile Networks[1]

This chapter describes localization in multi-hop mobile networks. Section 3.1 characterizes the patterns in real and synthetic mobility traces, Section 3.2 presents our approach and finally Section 3.3 describes our evaluation based on extensive trace driven simulations and testbed experiments.

3.1 Mobility Characterization

We first analyze the characteristics of mobility patterns using both real and synthetic traces, and find that they often exhibit low-rank structure and temporal stability.

3.1.1 Low-Rank Structure

Our first finding concerns the low-rank nature of the coordinate matrix. Specifically, consider n wireless nodes in a 2-dimensional Euclidean space. Let M be the $2n \times t_{\max}$ coordinate matrix over time, where $M(i,t)$ and $M(i+n,t)$ denote node i's x-coordinate and y-coordinate at time t ($1 \leq t \leq t_{\max}$),

[1]This chapter is revised version of the work in [82]. I was responsible overall for the project. My adviser Prof. Qiu and Prof. Zhang mentored and supervised the project. Yi-Chao helped me with the testbed evaluations.

Trace name	Description	Matrix size	Time interval
Cabspotting [18]	Taxis in Bay Area over a month	100 nodes × 300 intervals	1 minute
Shanghai taxi [121]	Taxis in Shanghai on 1 day	100 nodes × 300 intervals	1 minute
Seattle bus [89]	Buses in Seattle during 2001	545 nodes × 300 intervals	1 minute
ZebraNet [118]	ZebraNet deployment in 2005	61 nodes × 90 intervals	8 minutes
Human mobility [45]	Students' movement in KAIST	92 nodes × 499 intervals	30 seconds

Table 3.1: Real traces used in mobility characterization

respectively. It is not difficult to see that if every node moves at a constant velocity, even when different nodes move at different velocities, the coordinate matrix M always has rank 2. This is because we always have $M(:,t) = \mathbf{z} + t \cdot \mathbf{v}$, where $\mathbf{z}$ is a column vector that represents the initial coordinates of all the nodes, and $\mathbf{v}$ is a column vector that gives the velocities of all the nodes, and t is the current time. As a result, M can be represented as the sum of two rank-1 matrices: $M = \mathbf{z} \cdot \mathbf{1}^T + \mathbf{v} \cdot \mathbf{t}^T$, where $\mathbf{1}$ is an all-1 column vector, and $\mathbf{t} = [1, 2, \cdots, t_{\max}]^T$ is the time vector.

In practice, a node may not always move at a constant velocity. However, it is common that a node may travel at a constant velocity for some time before it changes direction or speed. This suggests that it is quite likely that the real coordinate matrix M exhibits low-rank structure. To validate this conjecture, we analyze a number of publicly available traces as shown in Table 3.1. Here the time interval is determined by the frequency of trace collection (*e.g.*, the vehicular traces recorded GPS readings of the vehicles at around once a minute). Since the human mobility traces and ZebraNet traces have only a few nodes, we pre-process these traces as follows. The KAIST campus traces were taken from 4 students living in a campus dormitory. There are 92 trace files, each of which represents a daily trace from one participant.

We treat each file as a trace from a distinct person, which gives us 92 nodes. We use ZebraNet traces from the second deployment, which has data from 5 zebras. We generate 61 synthetic zebras from these 5 real zebras by partitioning each trace over time and treating each partition as a synthetic zebra. This gives us 61 zebras over 90 time intervals. Since the original vehicular traces already contain a large number of nodes over an extended period, such pre-processing is not needed. Also note that some of these mobility traces contain a small fraction of missing values. We fill in these missing values using nearest-neighbor interpolation [108].

For comparison, we also generate synthetic mobility traces for 50 nodes over 100 intervals using the standard random waypoint model [16] and the modified random waypoint model proposed in [44]. The random waypoint model is one of the most widely used mobility models. In the standard random waypoint model, each node picks a random location as a destination and moves towards the destination at a randomly selected velocity; after reaching the destination, the node pauses for some random amount of time and selects a new destination and repeats the process. [44] proposes a modified random waypoint to overcome the limitation of the standard random waypoint model, which experiences decay in average speed, as reported in [114]. To maintain the average speed, in the modified random waypoint model, each node randomly chooses a speed every time interval (instead of staying at the same speed until reaching the destination as in the standard random waypoint model). We use maximum velocity of 10 meters/interval for low mobility and

30 meters/interval for high mobility under both standard and modified random waypoint models.

For each mobility trace, we first derive the corresponding coordinate matrix and mean center each row (*i.e.*, subtract from each row its mean value). We then apply singular value decomposition (SVD) to examine if the mean-centered coordinate matrix has a good low-rank approximation. The metric we use is the fraction of total variance captured by the top K singular values, *i.e.*, $\left(\sum_i^N s_i^2\right) / \left(\sum_i s_i^2\right)$, where s_i is the i-th largest singular value and $\left(\sum_i s_i^2\right)$ gives the total variance of the mean-centered coordinate matrix. Note that $1 - \left(\sum_i^N s_i^2\right) / \left(\sum_i s_i^2\right)$ is the relative approximation error of the best rank-K approximation with respect to the squared Frobenius norm. Figure 3.1 plots the fraction of total variance captured by the top K singular values for different mobility traces. As we can see, in all traces most variance is captured by the top few singular values. For example, the top 5 singular values capture 76.6%-94.9% variance in the real traces and 62.1%-97.2% variance in the synthetic traces. These results clearly suggest that real mobility exhibits low-rank structure.

3.1.2 Temporal Stability

We further analyze the temporal stability in these traces. For every node i and time interval t, we compute the normalized velocity change:

$$NVC(i,t) \triangleq \frac{\|\vec{v}(i,t) - \vec{v}(i,t-1)\|_2}{\text{mean}_{i,t}(\|\vec{v}(i,t)\|_2)},$$

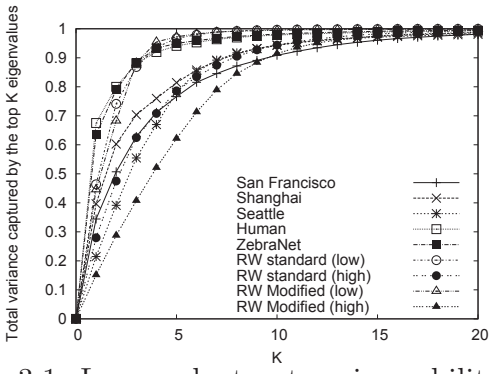


Figure 3.1: Low-rank structure in mobility traces

where $\vec{v}(i,t)$ is node i's velocity vector (in the 2-dimensional Euclidean space) at time interval t, and $\|\cdot\|_2$ is the ℓ_2-norm (with $\|\vec{z}\|_2 = \sqrt{\sum_k \vec{z}(k)^2}$ for any vector $\vec{z}$).

Figure 3.2 plots the CDF of $\{NVC(i,t)\}$. As it shows, four out of the five real traces have exactly the same velocity (*i.e.*, $NVC = 0$) in two consecutive intervals for over 36% of time, and NVC within 10% for over 42% of time. The only exception is Seattle bus traces likely due to frequent bus stops. Yet 34% of the time the bus traces show $NVC \leq 10\%$. In the standard random waypoint traces, 86%-93% of time $NVC \leq 10\%$. The modified random waypoint traces experience lower stability because every node chooses a random velocity at every time interval. These results indicate that the real mobility often exhibits temporal stability, *i.e.*, nodes move at constant velocity for some time. The extent of temporal stability varies with the network environment.

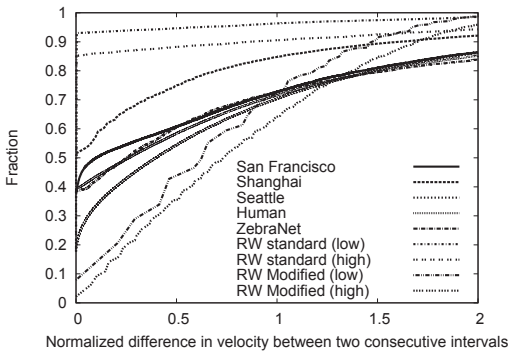


Figure 3.2: Temporal stability in mobility traces

3.1.3 Implications

The presence of low-rank structure and temporal stability in a wide range of mobility traces motivates us to explicitly exploit these structural properties to achieve better localization accuracy in mobile networks. Since the extent of low-rank structure and temporal stability may vary with the network environment, we cannot strictly enforce these properties in the localization solution. Instead we can incorporate them into the objective function. Specifically, static localization schemes aim to minimize the fitting error between observed and estimated distance based on nodes' coordinates. Now we can further minimize (i) the approximation error between the real coordinate matrix and its low-rank approximation, and (ii) the total change in velocity between two consecutive intervals.

3.2 Our Approach

3.2.1 Problem Formulation

In this section, we mathematically formulate the problem of mobile sensor localization. Specifically, we consider N mobile sensors moving in a d-dimensional Euclidean space. These mobile sensors have no built-in GPS and thus need to be localized by our localization algorithm. Meanwhile, we assume that there are N_{anch} mobile anchor nodes that are equipped with built-in GPS and thus have known locations. We divide time into $t_{\max}$ equal-sized time intervals. During each time interval, we measure (i) the locations for the anchor nodes, (ii) the distance between the sensors, and (iii) the distance between the sensors and anchors. We then try to localize the mobile sensors based on such measurements.

Notations: Our formulation uses the following notations:

- $X(:, i, t)$ denotes the d-dimensional Euclidean coordinate for sensor i ($1 \leq i \leq N$) at time interval t ($1 \leq t \leq t_{\max}$).
- $A(:, a, t)$ denotes the d-dimensional Euclidean coordinate for anchor a ($1 \leq a \leq N_{\text{anch}}$) at time interval t ($1 \leq t \leq t_{\max}$).
- $D_{ij}(t) = \|X(:, i, t) - X(:, j, t)\|_2^2 = \sum_{k=1}^{d}(X(k, i, t) - X(k, j, t))^2$ is the *squared Euclidean distance* between sensor i and sensor j at time interval t.
- $C_{ia}(t) = \|X(:, i, t) - A(:, a, t)\|_2^2 = \sum_{k=1}^{d}(X(k, i, t) - A(k, a, t))^2$ is the *squared Euclidean distance* between sensor i and anchor a at time interval

t.

- $D_{ij}^{\text{eq}}(t)$, $D_{ij}^{\text{ub}}(t)$, and $D_{ij}^{\text{lb}}(t)$ are the equality, upper bound, and lower bound constraints on $D_{ij}(t)$, respectively.
- $C_{ia}^{\text{eq}}(t)$, $C_{ia}^{\text{ub}}(t)$, and $C_{ia}^{\text{lb}}(t)$ are the equality, upper bound, and lower bound constraints on $C_{ia}(t)$, respectively.

Note that X is a 3-dimensional array. To formally capture the low-rank structure and temporal stability that we observe in Section 3.1, it is more convenient to represent X in the form of a 2-dimensional coordinate matrix, which can be obtained by collapsing the first two dimensions of X into a single dimension. Let $M = \mathsf{matricize}(X)$ be the resulted coordinate matrix. We have:

$$M(k + (i-1) * N, t) = X(k, i, t). \tag{3.1}$$

Incorporating distance measurements: We first formulate the localization problem for a static network by incorporating distance equality and bound constraints. We capture the total violation of distance constraints at time interval t using $f(X, t)$ shown below:

$$\begin{aligned} f(X,t) \triangleq\ & \textstyle\sum_{ij} \left(D_{ij}(t) - D_{ij}^{\text{eq}}(t)\right)^2 + \\ & \textstyle\sum_{ij} \min\left\{0, D_{ij}(t) - D_{ij}^{\text{lb}}(t)\right\}^2 + \\ & \textstyle\sum_{ij} \max\left\{0, D_{ij}(t) - D_{ij}^{\text{ub}}(t)\right\}^2 + \\ & \textstyle\sum_{ia} \left(C_{ia}(t) - C_{ia}^{\text{eq}}(t)\right)^2 + \\ & \textstyle\sum_{ia} \min\left\{0, C_{ia}(t) - C_{ia}^{\text{lb}}(t)\right\}^2 + \\ & \textstyle\sum_{ia} \max\left\{0, C_{ia}(t) - C_{ia}^{\text{ub}}(t)\right\}^2 \end{aligned} \tag{3.2}$$

Here the first and fourth terms quantify the total violation of the equality constraints $D_{ij}(t) = D_{ij}^{\text{eq}}(t)$ and $C_{ia}(t) = C_{ia}^{\text{eq}}(t)$. The second and fifth terms capture the total violation of the lower bound constraints $D_{ij}(t) \geq D_{ij}^{\text{lb}}(t)$ and $C_{ia}(t) \geq C_{ia}^{\text{lb}}(t)$. Similarly, the third and last terms represent the total violation of the upper bound constraints $D_{ij}(t) \leq D_{ij}^{\text{ub}}(t)$ and $C_{ia}(t) \leq C_{ia}^{\text{ub}}(t)$.

A few comments follow. First, we can use RSS measurements to derive equality constraints, because RSS is a function of the distance between the sender and the receiver. Moreover, when two nodes can directly hear each other, we can use the communication range as an upper bound of the distance between them. If two nodes cannot hear each other, then the communication range becomes a lower bound of the distance between them. In addition, if two nodes cannot directly hear each other but are connected through some indirect path, we can apply the triangular inequality and use the shortest path distance (in terms of the estimated distance based on RSS) as the upper bound. Note that all the equality and bound constraints on the Euclidean distance need to be squared, as $D_{ij}(t)$ and $C_{ia}(t)$ are defined with respect to the squared Euclidean distance.

Second, to support range-free localization, which is useful when the nodes can only get connectivity information but not the exact distance estimation, we simply remove equality constraints from $f(X, t)$ and only retain the upper bound and lower bound constraints. The lower bound constraints remain the same as above. As for the upper bound constraints, since the shortest path distance in terms of the estimated distance based on RSS is un-

available, we use a looser upper bound of $H \times R$, where H is the shortest hop count and R is the communication range.

Third, due to measurement errors, both the equality constraints and the upper/lower bound constraints may not be strictly satisfiable. By using the total violation against these constraints as our optimization objective (instead of trying to strictly enforcing these constraints), we can always find feasible solutions.

Fourth, it is easy to see that function $\max\{0, z\}^2$ is continuously differentiable with respect to variable z and the gradient is $2 \cdot \max\{0, z\}$. Similarly, function $\min\{0, z\}^2$ is continuously differentiable with respect to variable z and the gradient is $2 \cdot \min\{0, z\}$. Therefore, $f(X, t)$ is a continuously differentiable function of variables $X(:, i, t)$. This allows us to apply a gradient-based algorithm to minimize $f(X, t)$ (see Section 3.2.2).

Incorporating temporal stability constraints: To capture the temporal stability of sensor mobility, we introduce a temporal transformation matrix T and define a penalty function as follows:

$$g(X) \triangleq \left\| M * T^T \right\|_F^2, \tag{3.3}$$

where $\|\cdot\|_F$ is the Frobenius norm (with $\|Z\|_F = \sqrt{\sum_{ij} Z(i,j)^2}$ for any matrix Z), and $M = \mathsf{matricize}(X)$ is the coordinate matrix obtained by collapsing the first two dimensions of X into a single dimension according to Eq. (3.1).

A simple choice of the temporal transformation matrix is $T = Toeplitz(0, 1, -2, 1)$, which denotes the Toeplitz matrix with central diagonal given by ones, the first

upper diagonal given by negative twos, and the second upper diagonal given by ones, *i.e.*,

$$T = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots \\ 0 & 1 & -2 & 1 & \ddots \\ 0 & 0 & 1 & -2 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.4}$$

This temporal transformation matrix intuitively expresses the fact that the direction and the speed of the same mobile sensor are often similar at adjacent points in time. As a result, one can well approximate a sensor's location at time interval t using the mid-point of its locations at time intervals $t-1$ and $t+1$. With this T, we have:

$$g(X) = \sum_{k,i,t} \left(X(k,i,t-1) + X(k,i,t+1) - 2 \cdot X(k,i,t)\right)^2,$$

where each term $\left(X(k,i,t-1) + X(k,i,t+1) - 2 \cdot X(k,i,t)\right)^2$ represents the squared error between coordinate $X(k,i,t)$ and the mid-point of coordinates $X(k,i,t-1)$ and $X(k,i,t+1)$.

We use the above simple choice of T for mobile sensor localization in our evaluation. A more sophisticated choice taking into account domain knowledge of the nature of sensor movement is likely to further improve the localization accuracy. We intentionally go with the simple choice to better illustrate the importance of taking temporal stability into account.

Incorporating low-rank constraints: Finally, to capture the low-rank nature of sensor mobility, we introduce a penalty term function

$$h(X,U,V) \triangleq \left\|M - U * V^T\right\|_F^2 \tag{3.5}$$

where $M = \mathsf{matricize}(X)$ is the $(d \cdot N) \times t_{\max}$ coordinate matrix obtained by collapsing the first two dimensions of X according to Eq. (3.1), U is a $(d \cdot N) \times r$ unknown factor matrix, and V is a $t_{\max} \times r$ unknown factor matrix, and r is the desired low rank. By keeping this penalty term small, we ensure that the coordinate matrix M has a good rank-r approximation: $M \approx U * V^T$.

Complete formulation: Putting everything together, we therefore try to find X, U, V that minimize the combined objective:

$$c(X, U, V) = \sum_t f(X, t) + \alpha \cdot g(X) + \beta \cdot h(X, U, V), \tag{3.6}$$

where α and β give the relative weights of temporal stability and low-rank constraints, respectively. We show how to set α and β in Section 3.2.2.

Note that in Eq. (3.6), we can incorporate either or both of the penalty terms for temporal stability and low-rank structure (*i.e.*, $g(X)$ and $h(X, U, V)$). In this way, we can derive three different localization schemes: (i) Low Rank based Localization (LRL), which only exploits the low-rank structure in mobility (so $\alpha = 0$), (ii) Temporal Stability based Localization (TSL), which only leverages the temporal stability (so $\beta = 0$), and (iii) Temporal Stability and Low Rank based Localization (TSLRL), which simultaneously takes into account the temporal stability and the low-rank structure (so $\alpha \neq 0$ and $\beta \neq 0$). These localization schemes use distance estimation between neighbors and we call them *range-based* localization schemes. In addition, based on the general framework, we can also have *range-free* variants of these schemes, namely, LRL-RF, TSL-RF, TSLRL-RF. These variants use only network connectiv-

ity for localization and can support nodes that do not have accurate distance estimation (*e.g.*, due to lack of RSS measurements).

A couple of comments follow. First, the penalty term $\sum_t f(X,t)$, is the fitting error over all time intervals. This assumes that we have distance measurements from all intervals t. Our evaluation uses this assumption. When the distance measurements from some intervals are missing (*e.g.*, due to measurement problems or the need to predict nodes' coordinates in a future interval), we can simply use the sum of $f(X,t)$ over the intervals that have distance measurements as the first term in the objective. Second, $f(X,t)$ is a quartic function (*i.e.*, function of the fourth degree) with respect to X, whereas $g(X)$ and $h(X,U,V)$ are quadratic functions (*i.e.*, functions of the second degree) with respect to X. So there is a mismatch between the units of the different penalty terms in Eq. (3.6). To make the formulation independent of the unit we use for X, we can first normalize X such that the communication range equals 1. We assume the use of such normalization in the rest of the chapter.

3.2.2 Optimization

Optimization algorithm: We apply a quasi-Newton optimization algorithm L-BFGS [56] to find a local optimum of the above non-linear objective function $c(X,U,V)$. Quasi-Newton methods [110] are variants of the well-known Newton's method [109] for finding a stationary point (*i.e.*, a point where the gradient is 0) of a given objective function. Newton's method assumes that the objective function can be locally approximated as a quadratic function

in the region around the optimum, and uses the first and second derivatives (*i.e.*, gradient and Hessian) to find the stationary point. However, it is often expensive to directly compute the entire Hessian matrix for large optimization problems. To achieve better scalability, quasi-Newton methods do not compute the Hessian matrix directly. Instead, they update the Hessian matrix by analyzing successive gradient vectors.

L-BFGS stands for "limited memory BFGS". It is a particular quasi-Newton method that uses the Broyden-Fletcher-Goldfarb-Shanno update to approximate the Hessian matrix. L-BFGS only maintains a short history of the most recent m updates of the position (X, U, V) and gradient $\nabla c(X, U, V)$, where the history length m is typically less than 10. As a result, L-BFGS is particularly suited for optimization problems with a large number of variables.

We currently use the L-BFGS implementation in the *minFunc* package [67]. We set the history length $m = 5$, which provides good accuracy without compromising efficiency.

Choosing a good initial solution: Since L-BFGS is a gradient-based local search algorithm, it is critical to have a good initial solution. Without a good initial solution, the optimization tends to get stuck at local optima with poor localization accuracy, especially when d is small (*e.g.*, $d = 2$ or $d = 3$).

After extensive experiments, we find that the following initialization strategy often results in significant accuracy improvement:

1. First solve the problem in d'-dimensional Euclidean space using a random

initial solution, where $d' \geq d$ (*e.g.*, $d' = 4$ in our experiments). Let the solution be X''.

2. Compute the pair-wise distance matrix using X''.
3. Use multi-dimensional scaling (MDS) [24] to map the distance matrix obtained in step 2 into a solution in d-dimensional Euclidean space. Let the solution be X'.
4. Use X' as the initial solution and solve the original problem in d-dimensional Euclidean space.

Note that each point in the d-dimensional Euclidean space can be embedded into a d'-dimensional Euclidean space by setting its coordinates in the additional dimensions to 0. However, the converse is not true. That is, a point in a d'-dimensional space may not reside in a d dimensional space. Thus the d'-dimensional solution obtained in step 1 does not directly provide a solution for the original d-dimensional problem. That is why we apply MDS to project the solution back to the d-dimensional Euclidean space and use the result as an initial solution. Note that unlike existing MDS techniques in sensor localization [92, 93], we can directly apply the classic MDS [24], because the pair-wise distance matrix obtained in step 2 contains no missing entries.

The intuition behind the above initialization procedure is the following. When the dimension d is low, the feasible solution space for X may get partitioned into isolated islands. As a result, local search can easily get stuck at local optima with poor localization accuracy. In a higher dimensional Eu-

clidean space, due to the higher degree of freedom, different regions of the feasible solution space become better connected and it becomes harder to get stuck at a local optimum. As a result, a random starting point often suffices to yield a local optimum with high localization accuracy in the higher dimensional Euclidean space. Applying MDS to project this solution back to the low dimensional space can therefore yield a good initial solution for the original problem.

Our evaluation uses a single initial solution. One could also use multiple starting points and choose the one that yields the lowest error. This may further improve the localization accuracy at the expense of larger running time.

Tuning parameters α, β, and r: α and β control the importance of temporal stability and low-rank constraints, respectively. Their values depend on how noisy distance measurements are and how stable and low-rank the coordinate matrix is. When there is significant measurement noise, their values should increase to avoid being dominated by the large fitting error term. Moreover, α should also increase when the coordinate matrix exhibits strong temporal stability, and decrease otherwise. Therefore α depends on $\frac{\sum_t f(X,t)}{g(X)}$. To automatically adapt to diverse scenarios, we choose α using the following simple two iterations: we set $\alpha = 1$ in the first iteration and solve the optimization problem in Eq. 3.6; in the second iteration, we adapt α as follows: if $ratio > 1$, $\alpha = min(ratio, 10)$ where $ratio = \frac{\sum_t f(X,t)}{g(X)}$. We bound α by 10 to prevent it from being too large. Similar adaptation could be used in choosing

β. For simplicity, our evaluation uses a simple setting of $\beta = 0.1$, since we find it is more important to adapt α due to the higher importance of temporal stability constraints. Another parameter required in low rank constraints is the rank r. Our evaluation uses $r = 3$. As part of our future work, we plan to explore methods for automatically determining the appropriate rank r based on partial distance matrices.

Time complexity: As an iterative algorithm, the time complexity of L-BFGS depends on both the amount of time spent during each iteration and the number of iterations.

- *The time spent in each iteration.* In our context, the time spent during each iteration is dominated by the time for computing $\sum_t f(X,t)$ and $\sum_t \nabla f(X,t)$. As shown in Eq. (3.2), the expression for each $f(X,t)$ comprises $O((N+N_{\text{anch}})\cdot N)$ terms, each involving $O(d)$ variables. So it takes $O((N+N_{\text{anch}})\cdot N\cdot d)$ time to compute $f(X,t)$ and $\nabla f(X,t)$. Therefore, the total time complexity during each iteration is $O((N+N_{\text{anch}})\cdot N\cdot d\cdot t_{\max})$.
- *The number of iterations.* In our implementation, we simply run the L-BFGS algorithm for a fixed number of iterations (denoted as N_{iter}). Currently, we conservatively set $N_{\text{iter}} = 1000$, which suffices to ensure convergence in all our simulations and experiments. In our future work, we plan to incorporate less conservative convergence tests that allow the L-BFGS algorithm to terminate early.

Putting everything together, the time complexity for our current solu-

tion is given by $O((N + N_{\text{anch}}) \cdot N \cdot d \cdot t_{\max} \cdot N_{\text{iter}})$. The amortized cost per time interval is $O((N + N_{\text{anch}}) \cdot N \cdot d \cdot N_{\text{iter}})$.

3.3 Evaluation

This section describes the evaluation, both simulation and testbed.

3.3.1 Simulation

In this subsection we first describe our simulation methodology and then present the simulation results.

3.3.1.1 Simulation Methodology

We use a publicly available network simulator [64] for our evaluation. As in [63, 85], we quantify node density as the average number of nodes, including both regular nodes and anchor nodes, in one hop transmission range. It can be calculated as $\frac{\pi R^2 (N+N_{\text{anch}})}{area}$, where R is communication range, $area$ is the size of the deployment area, and N and N_{anch} are the total numbers of regular nodes and anchor nodes, respectively. Similarly, we quantify anchor density as the average number of anchors in one hop transmission range, calculated as $\frac{\pi R^2 N_{\text{anch}}}{area}$. Unless otherwise specified, we randomly place 50 nodes (including 5 anchor nodes) in a $200m \times 200m$ area, and use the following default parameter setting according to [62, 63]: node density of 10, anchor density of 1, communication range of 50 meters, and maximum speed of 10 meters/interval. Furthermore, since MCL and MSL* require warm-up period,

we use 30 intervals as input to the evaluation and quantify the localization error during the last 10 intervals. Without such warm-up period, the benefit of our localization schemes over MCL and MSL* is even higher. For each configuration, we conduct 10 random runs. In addition, we also vary each of these parameters to understand their impact.

Our evaluation uses both synthetic and real mobility traces. We generate synthetic node movement using the modified random waypoint model [44] (as described in Section 3.1) to overcome the limitation of speed reduction over time. We also try the standard random waypoint model and observe similar performance. Therefore for most evaluation, we report the results from the former mobility model in the interest of space. In addition, we also use the real mobility traces summarized in Table 3.1. In order to run 30 intervals across 10 random runs, we extract 300 consecutive intervals from these traces, except that ZebraNet has only 90 intervals so we conduct 3 random runs for the ZebraNet traces.

We compare our localization schemes with the following existing localization schemes: (i) Centroid [17], (ii) Multidimensional Scaling (MDS) [93], (iii) Sextant [39], (iv) MCL [44], and (v) MSL* [85]. The first three are well known localization schemes for static wireless networks and the last two are state-of-the-art localization schemes for mobile wireless networks. Refer to Chapter 2 for the description of these localization approaches.

We quantify the localization error using the mean absolute error (MAE), which has been widely used in previous studies. It is computed as $\mathsf{mean}_{i,t}(dist(X(:$

$, i, t), X'(:, i, t))$, where X and X' denote the actual and estimated coordinates, respectively, and $dist(X(:, i, t), X'(:, i, t))$ is the Euclidean distance between the actual and estimated coordinates for node i at time t.

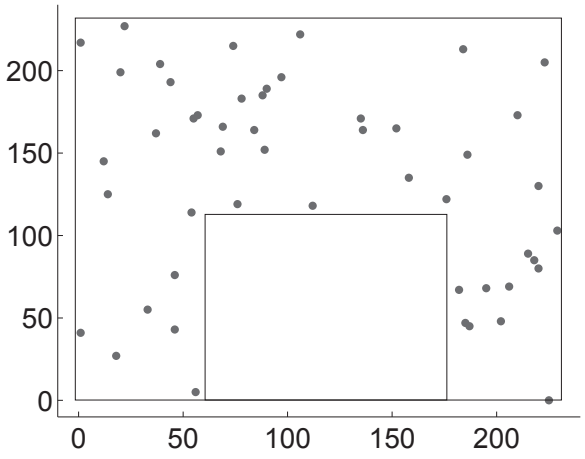
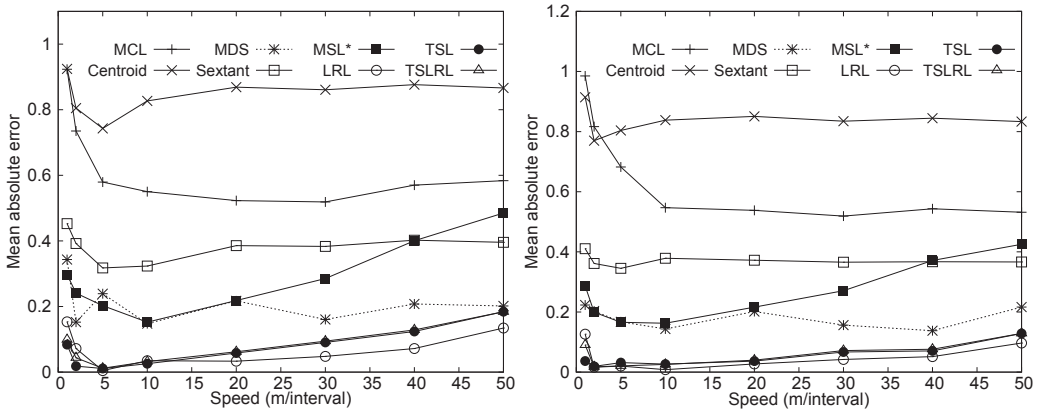


Figure 3.3: C-shaped area used for evaluation

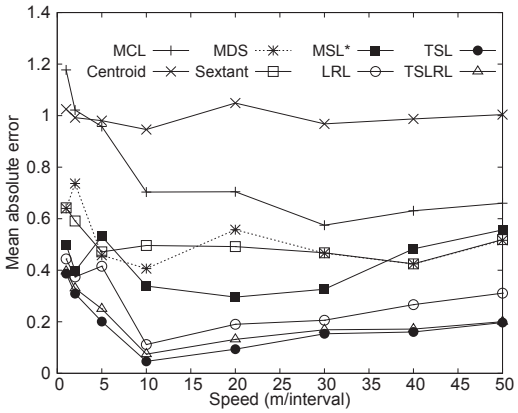
3.3.1.2 Simulation Results

Varying mobility: Figure 3.4(a) and (b) compare different localization schemes using the modified and standard random waypoint models, respectively. We make the following observations. First, in all cases, all our localization schemes significantly out-perform the other schemes. They out-perform Centroid by 50%-90%, MDS by 10%-80%, Sextant by 30%-90%, MCL by 30%-90%, and MSL* by 20%-80%. Second, the localization error in the modified mobility model is slightly higher than that in the standard mobility model due to more frequent change in velocity, but their general trends and relative performance across different schemes are very similar. Therefore we use the



(a) Modified random waypoint

(b) Standard random waypoint



(c) Standard random waypoint in a C-shaped area

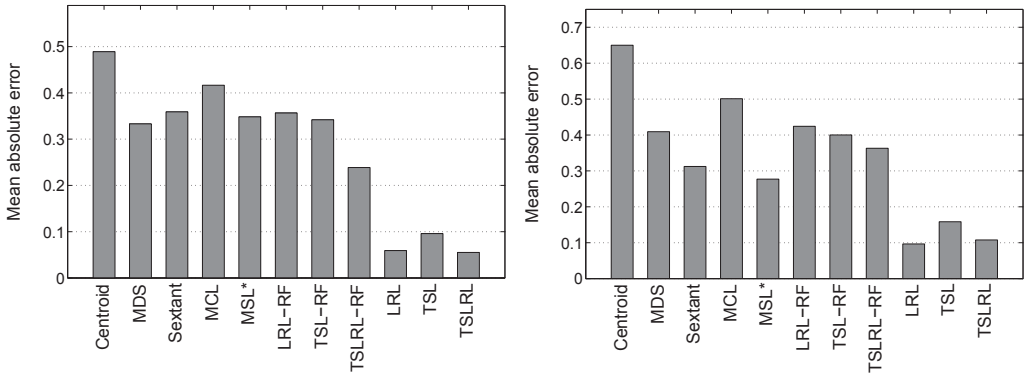
Figure 3.4: Localization accuracy under random waypoint models

modified random waypoint for the rest of our evaluation since it is used in [63, 85]. Third, the errors of all mobile network localization schemes initially decrease as we increase mobility because at very low mobility the network topologies change little and makes it hard to extract new information. The error then increases slightly with increasing mobility as the low-rank structure and temporal stability reduces slightly with increasing mobility. This behavior

is more prominent in the modified random way point model, as the node selects a new speed every interval, whereas in the standard random way point model the speed remains constant for consecutive intervals until the node reaches its destination.

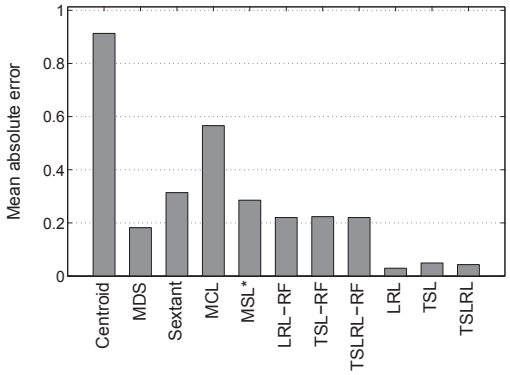
We also evaluate using an irregular C-shaped area shown in Figure 3.3. It has a 230×230 square with a 115×115 block taken off. We generate nodes' coordinates over time using the standard random way point mobility within the area. To ensure every node's position falls within the C-shaped region, whenever it is about to traverse outside the region, we re-select a new destination and move towards it at a randomly selected speed. As shown in Figure 3.4(c), the results are similar to before, except that the accuracy of MDS degrades and sometimes under-performs MSL* and Sextant. This is a well known issue with MDS, because in irregular topologies the true Euclidean distance between two nodes does not correlate well with the shortest path distance, which is used as the input to MDS. In comparison, our localization schemes only use the distance information between direct neighbors as equality constraints and are less sensitive to the above issue. As a result, they continue to perform the best.

We further evaluate using real mobility traces. We pick 100 vehicles randomly from the vehicular traces and use all the nodes from the KAIST campus traces and ZebraNet traces. For all the traces, we scale down the distance. Figure 3.5 summarizes the results. As we can see, our range-based localization schemes consistently yield much lower errors than the other schemes across all

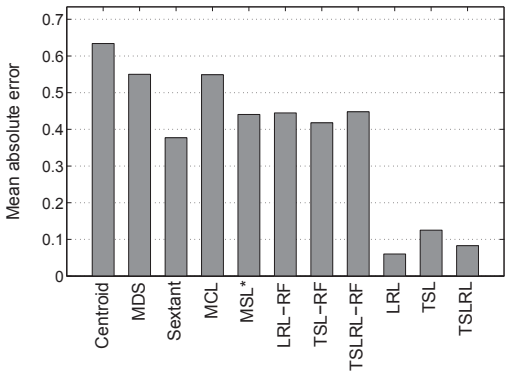


(a) San Francisco taxis

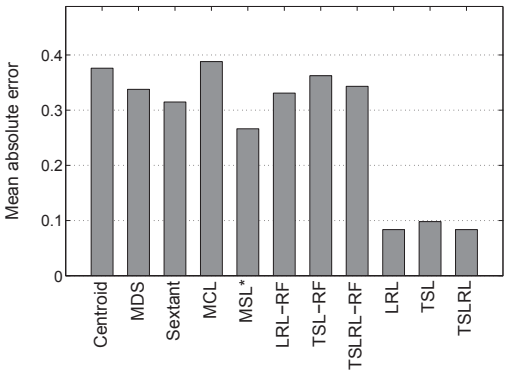
(b) Seattle bus



(c) Shanghai taxis



(d) ZebraNet



(e) Human mobility

Figure 3.5: Localization accuracy with real traces

the traces. The range-free versions, namely, LRL-RF, TSL-RF, and TSLRL-RF, perform worse than their corresponding range-based counterparts due to lack of exact distance information. Among the existing schemes, MDS and MSL* perform better. However, their accuracy is still significantly worse than our range-based schemes since they do not fully exploit topology and mobility information.

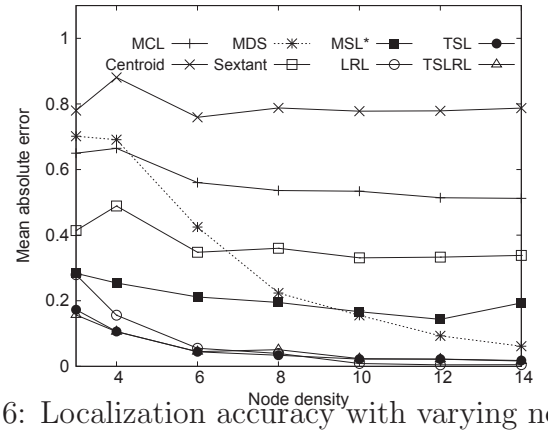


Figure 3.6: Localization accuracy with varying node density

Varying node density: Figure 3.6 compares various localization schemes under varying node density. As it shows, our localization schemes continue to perform the best under all node density. In addition, all the localization schemes have lower error with increasing node density. The only exception is Centroid, which sees similar error over all node density. This is because Centroid extracts location information only from anchor nodes and does not exploit the relative distance information between regular nodes. Therefore its accuracy is determined only by the anchor nodes and benefits little from higher density of regular nodes. Similar effects were reported in [44].

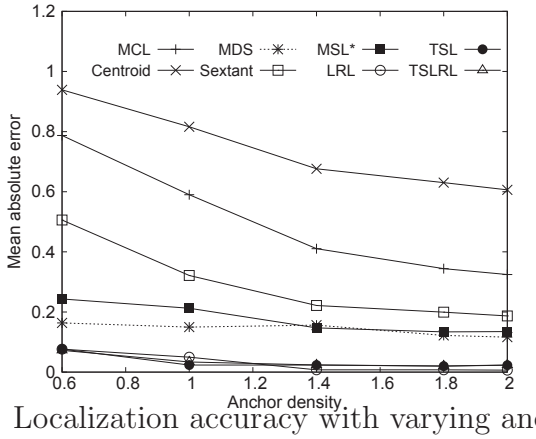
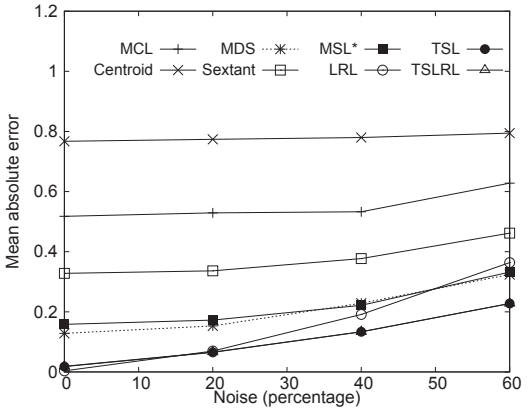


Figure 3.7: Localization accuracy with varying anchor density

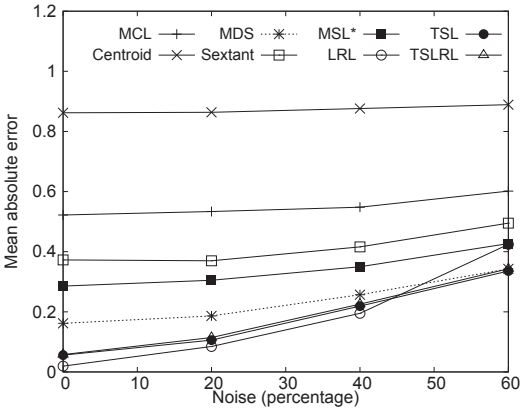
Effect of anchor density: We further vary the anchor density. Figure 3.7 plots the localization errors as the anchor density varies. As we would expect, all localization schemes benefit from increasing anchor density. Moreover, our localization schemes remain the best.

Varying noise: It is important to study the accuracy of all the schemes under measurement noise, since noise is common in real deployments. Measurement noise comes from two factors: (i) there is irregularity in transmission range, *i.e.*, not all nodes have the same transmission range and even for the same node its transmission range is not the same in all directions, and (ii) a neighboring node may estimate inaccurate distance based on its RSS measurements. To capture the first effect, we generate a random number r_{ij} uniformly distributed from $[1 - noise, 1 + noise]$ and we vary $noise$ from 0 to 60%. The new transmission range between the node pair becomes $r_{ij} \cdot R$. To capture the second effect, we perturb the distance estimation as d_{ij}/r_{ij}, where d_{ij} is the actual distance between i and j and r_{ij} is the scaling factor used to generate noisy

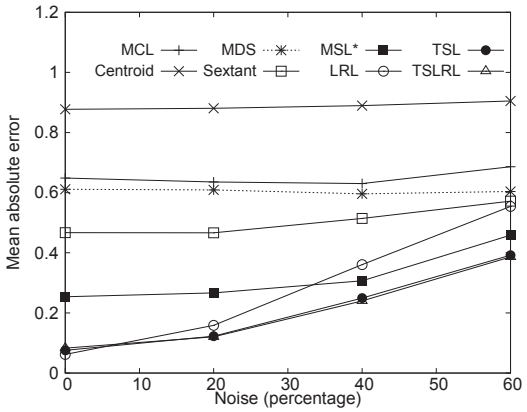
transmission range between i and j. In this way, we ensure the error added to transmission range is consistent to the error added to distance estimation. We then drop the entry (i, j) in the distance matrix if the actual distance between i and j is larger than the salted transmission range (since they are not direct neighbors and cannot measure RSS), and input this distance matrix to all localization schemes.



(a) Node density = 10, max speed = 10 m/interval

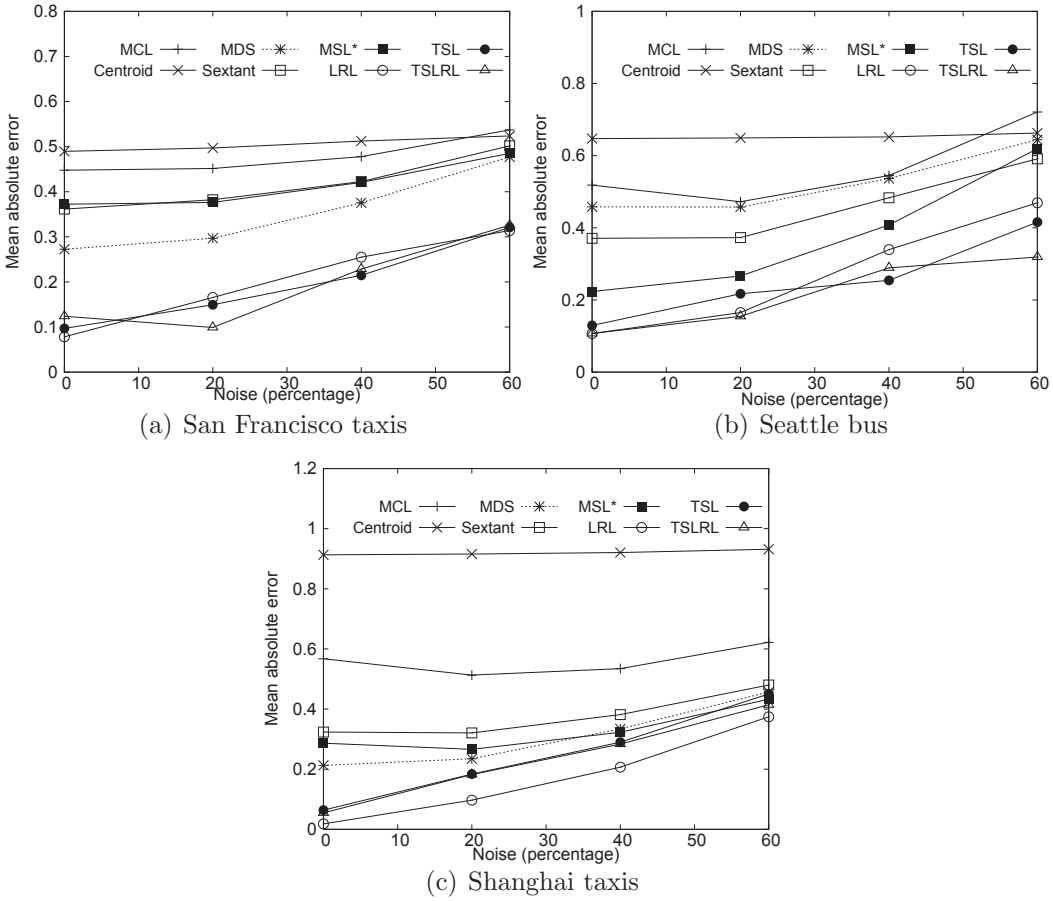


(b) Node density = 10, max speed = 30 m/interval



(c) Node density = 4, max speed = 10 m/interval

Figure 3.8: Varying noise in 2-D networks



(a) San Francisco taxis

(b) Seattle bus

(c) Shanghai taxis

Figure 3.9: Varying noise in the real vehicular traces

Figure 3.8 compares various localization schemes by varying the amount of noise. Figure 3.8(a) shows the results under the default network configuration, Figure 3.8(b) shows the results under higher mobility, and Figure 3.8 (c) shows the results under lower node density. We make the following observations. First, as we would expect, increasing noise degrades the accuracy of all the localization schemes. Among them, the error in Centroid increases slowest with increasing noise, because it estimates its location as the center of

all anchor nodes it hears from and is not affected significantly by measurement errors. Similar effects were observed in [44]. Second, TSL and TSLRL continue to yield the lowest errors under all noise, whereas LRL performs slightly worse. LRL initially out-performs all the existing schemes under low noise but then slightly under-performs the best of the existing schemes under high noise. This is likely because increasing noise may affect low rank structure and reduce the effectiveness of low rank constraints. In comparison, temporal stability is more robust to noise so that TSL and TSLRL perform considerably better than both MDS and LRL. Fourth, increasing the maximum speed from 10 to 30 slightly degrades the accuracy of various schemes. Among them, MSL* is affected the most, because it uses the maximum speed to generate feasible node positions during the next intervals and location uncertainty increases with mobility. Finally, as we would expect, the accuracy of all the schemes degrades as the node density decreases due to fewer location constraints.

Figure 3.9 evaluates the impact of noise on real traces. As in the synthetic traces, increasing noise degrades the accuracy of all localization schemes. Moreover, our localization schemes continue to perform the best under all noise values.

To demonstrate the flexibility of our scheme, we also consider varying noise when nodes have 3-D coordinates. We place 50 nodes including 5 anchor nodes randomly in a cube with each side of 140 meters. Figure 3.10 compares our localization schemes with the other schemes except Sextant, which works only in 2-D. We observe that TSL and TSLRL perform similarly and their

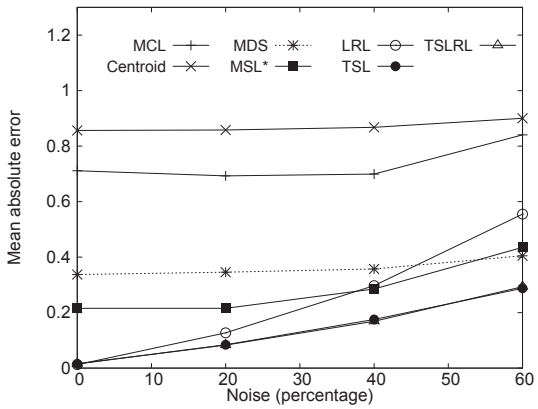


Figure 3.10: Varying noise in 3-D networks

curves overlap. They both yield the lowest error in all the cases. In comparison, Centroid yields the highest error since it only uses anchor nodes for localization, which gives very limited location constraints.

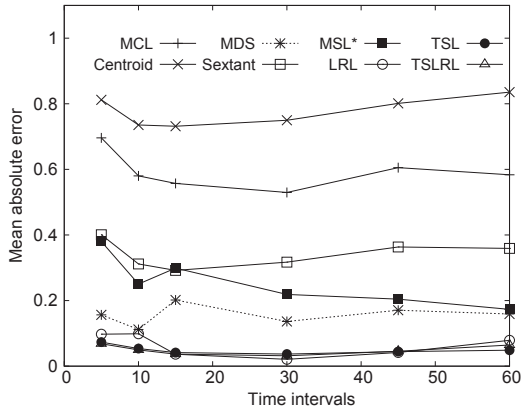


Figure 3.11: Impact of the number of time intervals

Varying the number of time intervals: Figure 3.11 shows the localization error as we vary the number of intervals. As MCL and MSL* require warm-up period, we use 5 warm-up intervals and vary the number of intervals for localization from 5 to 60. As we can see, TSL, LRL and TSLRL yield the lowest

errors under all numbers of intervals. In comparison, MCL incurs mean absolute error of 0.6 even with 60 intervals. MSL* improves the convergence over MCL by letting nodes use information only from the neighbors that have more accurate coordinates, and reduces mean absolute error to 0.2-0.4. However, it still under-performs our schemes because it does not fully exploit location constraints during each time interval nor the structure in mobility, other than the maximum speed.

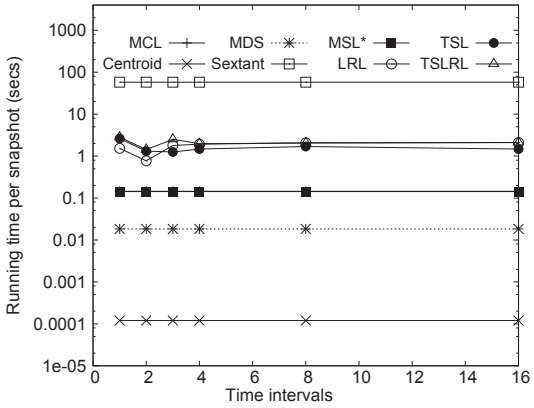


Figure 3.12: Running time

Running time: Finally we compare the running time of different approaches on a Linux machine with Intel(R) Core(TM)2 Duo CPU E8200 2.66GHz processor, 2GB memory, and 6 MB cache. Figure 3.12 shows the average running time per interval as we vary the number of intervals for the default configuration (*i.e.*, 50 nodes including 5 anchor nodes, node density of 10, anchor density of 1, and maximum speed of 10). Our schemes (*i.e.*, LRL, TSL, TSLRL) and MDS are implemented in Matlab, while the other schemes were implemented in Java by other researchers. Java implementation is generally faster than

Matlab. As shown in Figure 3.12, all of the schemes have close to constant running time per interval. In other words, their total running time increases linearly with the number of intervals. The ranking of their running time is as follows: Centroid $<$ MDS $<$ MSL* $\approx$ MCL $<$ LRL $\approx$ TSL $\approx$ TSLRL $<$ Sextant. Sextant has highest running time, around 1 minute per interval, because computing the regions that satisfy all the distance measurement constraints is expensive. Our approaches, LRL, TSL, and TSLRL, take 1-2 seconds per interval. This is longer than the existing schemes except Sextant because we try to fully exploit the location information within each interval and mobility structure across intervals. However this running time is affordable for practical use. Moreover, our implementation has room for optimization (*e.g.*, adaptively choosing the number of iterations as mentioned in Section 3.2.2 and converting Matlab to C implementation).

Summary: Our simulation results show that our localization schemes significantly out-perform the existing schemes under a wide range of scenarios. They are also robust to measurement errors and affordable for practical use.

3.3.2 Testbed Experiments

3.3.2.1 Experimental Methodology

In this section, we evaluate our localization scheme in a sensor testbed. We deploy a testbed consisting of either 25 or 36 *mica*2 motes with 915MHz radios on a single floor of an office building. This allows us to evaluate our localization algorithm under realistic radio characteristics. In our first exper-

iment, we randomly place them in a square with $5.5m \times 5.5m$. This gives node density of 20. We use the standard random waypoint [44] to generate the motes' coordinates. We use two types of mobility: low mobility with maximum speed of 0.2R and high mobility with maximum speed of 0.6R. In our second experiment, we place the motes in a C shaped topology shown in Figure 3.14(a), occupying a total area of $6.5m \times 6.5m$, with a smaller square of $3.4m \times 3.4m$ taken away. It has a node density of 15. We again use the standard random waypoint to generate the coordinates in the subsequent time intervals while ensuring that the motes move within the region. For diversity, we use medium mobility with maximum speed of 0.4R. In both topologies, we use 15 time intervals and move these motes by hand at the beginning of each interval to ensure the actual locations of the motes correspond to the generated locations.

Among these motes, we randomly choose 6 anchor nodes in both topologies. We adjust the transmission power to $-24dBm$ for all control and data traffic. This gives the communication range of about 2.3 meters, and yields multi-hop topologies with up to 3 hops. One mote, referred as the sink, is attached to the MIB600 Ethernet board, which is connected to a power outlet. We use the sink to log all the measurement data. The other motes are powered by the batteries.

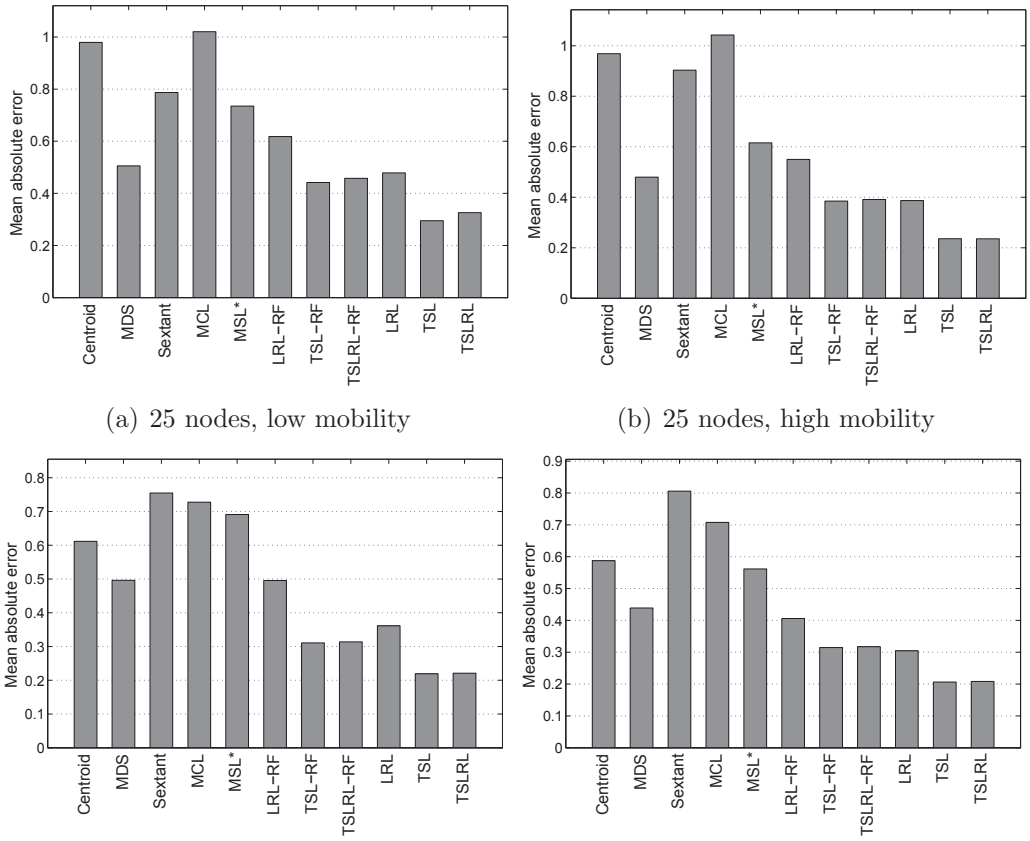
During each time interval, the sink first broadcasts a route establishment frame (REF), which is flooded throughout the network so that every other node can route towards the sink by reversing the shortest path along

which REF traversed. To improve the reliability of the route, every node selects the path with the smallest ETX to route towards the sink. ETX quantifies the total number of expected transmissions from a source to a destination [26]. To support such a route selection, an intermediate node i appends the ETX between the sink and node i to REF before forwarding.

Next every mote broadcasts several packets at a random time so that its neighbors can measure the received signal strength from it. Each mote broadcasts once every 1.5 seconds, and chooses a random waiting time between 0.1 and 1.5 seconds to broadcast every broadcast interval in order to minimize collisions among different nodes' broadcast packets. Every mote periodically sends a report to the sink to summarize the measured received signal strength from all nodes it hears during the current period. The report is routed using the shortest ETX path selected above. We then take measurements collected from the sink over 15 intervals and use them to evaluate the performance of various localization schemes. We estimate the distance between neighboring nodes during a given time interval using average RSS of all the packets received during that interval.

3.3.2.2 Experimental Results

Figure 3.13 shows the mean absolute error for 25-node and 36-node networks in the first topology. We make the following observations. First, all our range-based localization schemes significantly out-perform the other schemes. In particular, TSL and TSLRL improve Centroid by 65%-75%, MDS by 40%-



(a) 25 nodes, low mobility

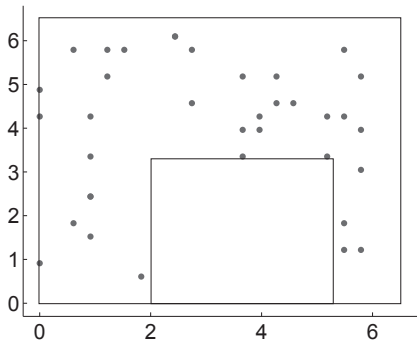
(b) 25 nodes, high mobility

(c) 36 nodes, low mobility

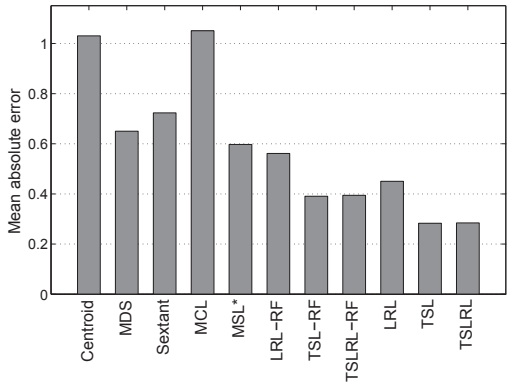
(d) 36 nodes, high mobility

Figure 3.13: Localization accuracy: testbed in square region

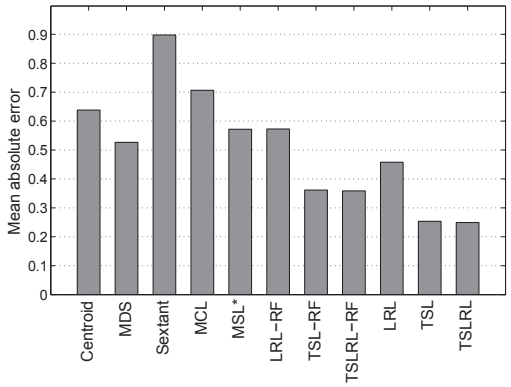
55%, Sextant by 60%-75%, MCL by 70%-80%, MSL* by 60%-70%. Second, our range-free versions also perform well: TSL-RF and TSLRL-RF both yield lower errors than the existing schemes. LRL-RF occasionally performs worse than MDS, the best of the existing schemes, because it only exploits low rank structure, which may be affected by measurement noise in the testbed. Third, the gap between the range free and range based versions becomes smaller than that in simulation. This is because the testbed experiments have significant errors in estimating the distances based on RSS measurements. This



(a) C-shaped topology used in experiments



(b) 25 nodes, irregular topologies



(c) 36 nodes, irregular topologies

Figure 3.14: Localization accuracy: testbed in C-shaped region

is further confirmed by Figure 3.15, which plots the CDF of the normalized difference between the true distance matrix and the estimated distance matrix based on RSS measurement. Such measurement noise makes the range based constraints noisy. Fourth, as in simulation, TSL and TSLRL perform similarly and out-perform the others, which indicates that temporal stability is important for mobile network localization and robust to measurement noise. Finally, comparing the five existing localization schemes, MDS performs the best among the static localization schemes, and MSL* significantly improves

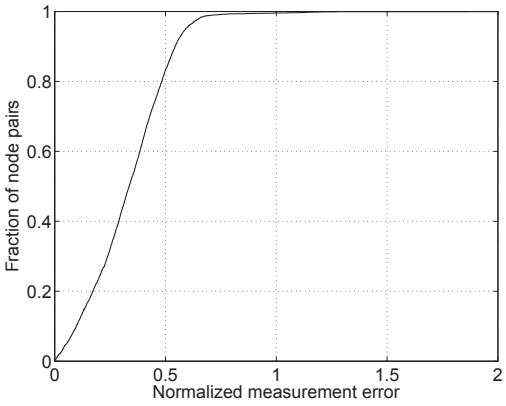


Figure 3.15: CDF of measurement error in the testbed

accuracy of MCL in mobile network localization due to its faster convergence. Even though MSL* leverages mobility information, it still does not perform as well as MDS because it does not fully take advantage of network topology at each time interval. By simultaneously exploiting the complete topology information at individual intervals and the structure in mobility, our approaches achieve much better accuracy.

Figure 3.14 (b) and (c) further plot the localization errors in a 25-node and 36-node network deployed in a C-shaped region, as shown in Figure 3.14(a). Both range-free and range-based versions of our localization schemes out-perform the existing schemes. The range-based versions of TSL and TSLRL continue to perform the best, out-performing Centroid by 60%-70%, MDS by 50%-55%, Sextant by 60%-70%, MCL by 65%-70%, MSL* by 50% - 55%. In addition, as we would expect, the accuracy of MDS degrades in irregular topologies due to lack of strong correlation between the shortest path distance and Euclidean distance, as described in Section 3.3.1.2.

Summary: Our testbed experiments show that our localization schemes out-perform the existing schemes in a range of settings. Among them, the range-based versions of TSL and TSLRL consistently perform the best due to the effectiveness and robustness of temporal stability constraints.

Chapter 4

Localization for Single Mobile Nodes[1]

This chapter describes localization of a single mobile node, which may not be connected with multiple nodes (e.g., without network connectivity or only connected with an access point). Section 4.1 motivates our approach, Section 4.2 presents our approach and finally Section 4.3 describes the evaluation.

4.1 Characterizing Location Signatures

We first introduce various types of location signatures. We then motivate the need for trajectory based localization by identifying the limitations of point based localization and showing the advantages of trajectory based localization.

4.1.1 Location Signatures

This subsection describes different types of location signatures.

[1]Part of this chapter appeared in [80]. I was responsible overall for the project. Wei spent a lot of time brain-storming with me and also helped collecting traces. My adviser Prof. Qiu and Prof. Zhang mentored and supervised the project.

4.1.1.1 Magnetic Field

The earth's magnetic field is a ubiquitous signature that can also potentially be used for localization. The magnitude of the field from the earth alone varies between $0.3 - 0.6$ Gauss $(30 - 60\ \mu T)$ [33]. However, the earth's magnetic field can be affected by many other factors, such as speakers, electric lines, appliances, etc. The impact of these factors differs across locations. This gives an opportunity to use magnetic field as a location signature.

4.1.1.2 Wi-Fi signals

A widely used location signature is Received Signal Strength (RSS), since RSS has a direct relationship with the distance from the transmitters. If the relationship between RSS and distance is known (*e.g.*, in free-space), we can use RSS from three transmitters to uniquely determine the location. However, in practice, the relationship between RSS and distance is much more complex due to multipath effects, interference, and noise. This significantly limits the accuracy of RSS-based localization.

Recognizing the limitation of RSS-based approach, more recent works [75, 119, 91] leverage the more fine-grained Channel State Information (CSI) to improve the localization accuracy. CSI gives signal-to-noise (SNR) information for every OFDM subcarrier (or subcarrier group). When N transmitting antennas send to M receiving antennas, the CSI consists of $M \times N$ matrices, where a matrix gives the amplitude and phase between a pair of transmitting and receiving antenna on a subcarrier. For example, the Intel Wi-Fi

Link 5300 (iwl5300) 802.11 a/b/g/n wireless network adapters that we use report the channel matrices for 30 subcarrier groups (around 2 subcarriers per group). This more fine-grained signature improves accuracy, because different locations are much less likely to have the same CSI than RSS.

4.1.2 Point-based Localization: Limitations

Point-based localization schemes determine a location based on measurements collected from that individual location or point.

4.1.2.1 Magnetic Field

Here, we demonstrate that the accuracy of localization using magnetic field readings from a single point is poor. We evaluate point-based localization using the magnetic field outdoors. We use the drive traces that we collect as described in Section 4.3. We pick every 100-th point from lap 2 for testing and try to match these points with the points in lap 1 based on the closest magnetic field value. We evaluate both with and without the knowledge of the specific segment. Table 4.1 shows the results. We can see that the accuracy is very poor. Without the segment information the error is hundreds of meters and with the segment information tens of meters. This is due to significant aliasing as points far apart can have similar magnetic field magnitude.

Trace	# Points		Error	
	Total	Per Seg.	Overall	Per Seg.
1	152	15	199m	54m
2	143	14	278m	51m
3	146	15	229m	45m
4	141	14	259m	45m
5	148	15	231m	55m
6	185	11	318m	62m
7	121	9	158m	48m
8	128	13	211m	49m
9	104	6	161m	22m

Table 4.1: Accuracy of point based localization outdoors

4.1.2.2 Wi-Fi Signals

PinLoc [91] uses CSI for localization. As the authors point out, CSI from locations even just $2cms$ away can be very different. So PinLoc uses significant training data by having a roomba robot moving in a $1m$ by $1m$ grid for 4 minutes collecting the training data which is an average of 60 samples from each $2cm$ by $2cm$ grid in order to identify which grid a new measurement falls into. They further cluster the training data and localize a user to a $1m$ by $1m$ spot if multiple measurements are classified as falling into that spot.

To understand the performance of point based localization using CSI with sparse training data, we collect a CSI trace using the Intel 5300 cards based on the tool developed in [40]. We measure the CSI in a regular office building, by having 44 grids each measuring $2m$ by $2m$. We use MCS 0, and 15 dBm Tx power for the measurement. We collect the CSI from 5 senders at 3 points in each grid. This gives us altogether 132 locations. Each location collects $30,000$ packets. Each packet has CSI values from 30 subcarrier groups

on each pair of Tx and Rx antennas, which gives us a total of 90 values on all 3 receiving antennas.

We quantify the performance based on the following information in the CSI: (i) phase, (ii) amplitude, and (iii) combination of phase and amplitude. The initial phase of each transmission varies randomly. To get more reliable phase signature, we post-process the raw phase information to remove the random initial phase. Whenever $phase(s+1) < phase(s)$, where s is the subcarrier index, we update $phase(s+1) = phase(s+1) + 2\pi$, since the phase of subcarriers in a frame should change monotonically. We then remove the random initial phase of the first subcarrier from the phases of all the subcarriers, *i.e.*, $phase(s) = phase(s) - phase(1), \forall s$. We compute the mean CSI for each subcarrier group over all packets, which gives us a vector of 3 $(antennas) \times 30$ $(subcarrier\_groups) \times 5$ $(senders) = 450$ mean phase values and use this as the signature of localization. (ii) Amplitude signature is simply a vector of 450 values of the mean amplitude of each subcarrier group across all packets. (iii) Combined signature: for the CSI measurement on subcarrier s, once we get the corrected phase, $phase'(s)$, we compute the new complex CSI value: $a' + b'i$, with $a' = A(s) \times cos(phase'(s))$ and $b' = A(s) \times sin(phase'(s))$, where $A(s)$ is amplitude of the signal at subcarrier s. We obtain the combined signatures as a vector of 450 complex values, which represent the mean over all packets.

We first evaluate the performance of localization when training data from the exact location is available. We divide the $30,000$ packets from each

location into a training trace of $20,000$ packets and a testing trace of $10,000$ packets. We then find the best match for each of the locations using the testing trace. 132/132 locations are matched correctly while using phase signature and amplitude signature and 131/132 locations are matched correctly using the combined signature. On closer inspection of the one incorrect match from the combined signature, we find that the right match is still within top 5 closest matches with small differences in the signature. This shows that CSI from the exact same location is likely to be similar at least over small time-scales, as in this experiment, where training and test traces were collected back-to-back.

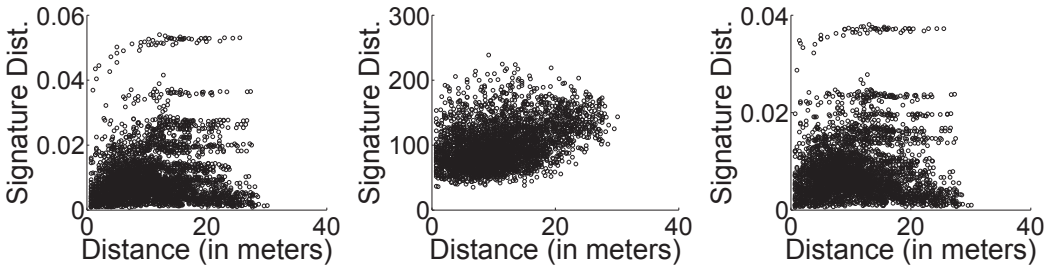
Next we evaluate a more realistic case when there is no training data from the exactly same points and check if we are able to match the signatures to the closest location. We take one location from each grid of size $2m$ by $2m$ as the training location, so altogether we have a total of 44 training locations, and 2 locations from each grid of the same size as the testing locations, which gives a total of 88 test locations. We deem correctly classified if we match to the closest location, among the closest 2 locations, or among the closest 5 locations. As shown in Table 4.2, we match to closest location only 24% of the the time using amplitude signature, 9% of the time using phase signature, and 11% of the time using combined signature. The limited accuracy is because CSI at nearby locations can be very different. Specifically, the wavelength of the Wi-Fi signals is about $5.8cms$ for channel 36, 5.18 GHz, and phase is reversed every half wavelength, so even very nearby points can have very different phases. Amplitude information is robust against phase variations but

Error method	Amplitude	Phase	Combined
Dist. error	$5m$	$7m$	$7.8m$
Match to top 1	24%	9%	11%
Match to top 2	41%	13%	18%
Match to top 5	65%	35%	41%

Table 4.2: Accuracy of matching to neighboring locations

still performs poorly due to measurement noise and poor correlation between geographic distance and signal strength.

Figures 4.1 (a), (b), and (c) further show the correlation between the real distance between the locations and the distance between their corresponding amplitude, phase, and combined signatures, respectively. We see that as the real geographic distance increases, the difference between the signature distance may not always increase. This explains the poor accuracy of matching to the closest locations.



(a) Amplitude (b) Phase (c) Combined

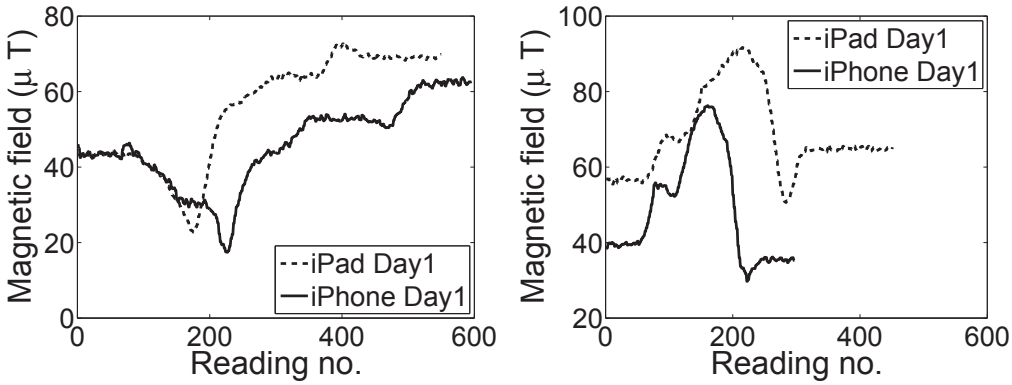
Figure 4.1: Real distance vs. signature distance

4.1.3 Potential of Trajectory-based Localization

In this section, we study the potential of trajectory based localization. Trajectory based localization has the potential to significantly out-perform the point-based localization schemes by leveraging information across multiple

points along the trajectory.

4.1.3.1 Magnetic Field

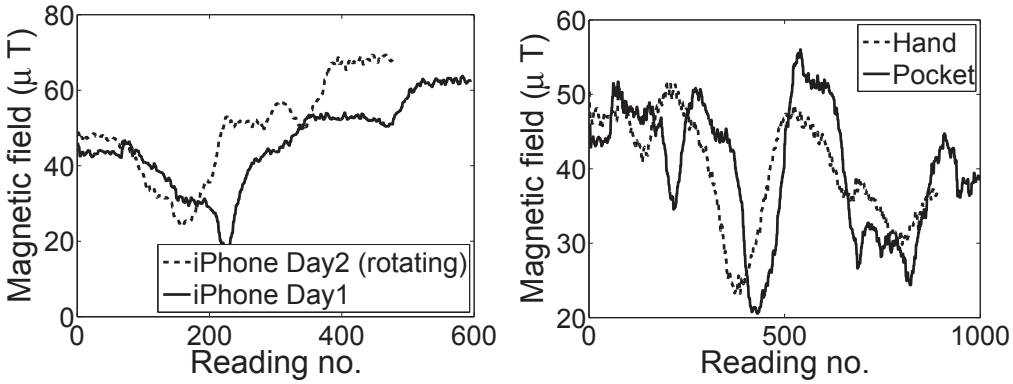


(a) Location 1 to 2 (b) Location 3 to 4

Figure 4.2: Magnetic field, different devices

We measure the magnetic field along a few trajectories under different conditions. Figure 4.2 (a) and (b) compare the magnetic field between the two points on an indoor trajectory (of length 1.5m) measured using different devices (iPad vs. iPhone). While the curves have different shift and scale since the devices have different sensitivity and different speeds, their patterns look similar.

Figure 4.3 (a) shows the magnetic field collected using an iPhone on two different days. To understand the impact of device orientation, we recorded one trace by placing the iPhone on a book and rotating it while pushing the book from Location 1 to Location 2 (they are 1.5 m apart). While all other traces were taken by simply pushing the book. Further, in Figure 4.3 (b), we



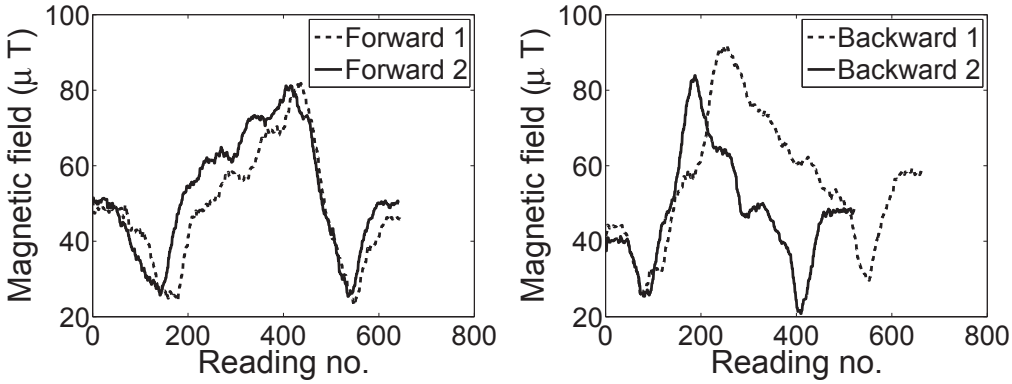
(a) Different day, orientation (b) Hand vs. Pocket

Figure 4.3: Magnetic field across same trajectory

see that two different positions of the phone – hold it in hand and walk or put it in pocket, still yield similar magnetic field pattern (on a 7.5m long indoor trajectory).

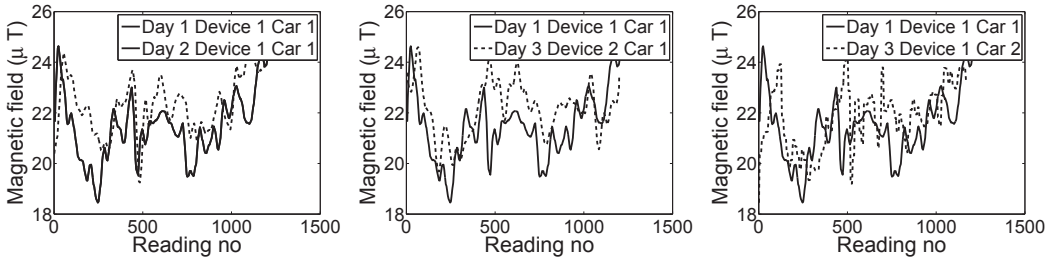
To understand the impact of walking direction, we walk from location 1 to location 2 and then walk back from location 2 to location 1. As Figure 4.4 (a) and (b) show, the measurements collected in the same direction look similar, while the measurements collected in the opposite directions look reversed. This shows that it is sufficient to have the training traces in one direction and to have them reversed for the opposite direction.

Next we look at magnetic field pattern outdoors while driving. Several factors may affect the magnetic field patterns outdoors, such as measurement time, types of devices, and cars. Figure 4.5 (a) shows the magnetic field pattern on a street in Redmond Town Center on two different days, and the pattern remains reasonably similar. Figure 4.5 (b) shows the pattern across different



(a) Forward (b) Backward

Figure 4.4: Magnetic field patterns while walking forward and backward



(a) Different Days (b) Different Devices (c) Different Cars

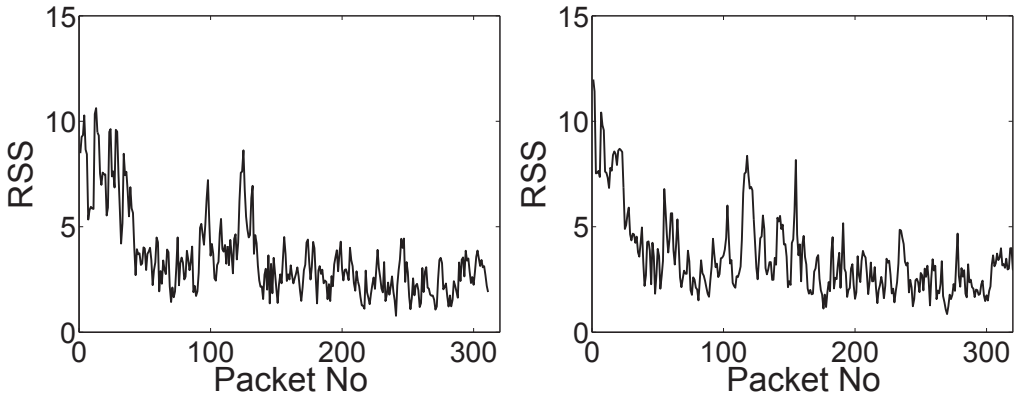
Figure 4.5: Outdoor magnetic field pattern

devices is similar. Figure 4.5 (c) further shows the pattern across different cars remains the same. We further show in Section 4.3 that magnetic field can be leveraged to perform localization with errors lower than a meter indoors, and close to GPS accuracy outdoors.

4.1.3.2 Wi-Fi Signals

Here, we show that Wi-Fi signal trajectory can be leveraged for localization too. In Figure 4.6, we plot the RSS over the same 4m long indoor

trajectory taken in two different walks and see that the pattern remains similar. In Section 4.3, we will show that using RSS and CSI information across trajectory yields accurate localization with errors close to about 0.3m.



(a) Lap 1 (b) Lap 2

Figure 4.6: RSS time-series

4.2 Our Approach

4.2.1 Overview

Motivated by the potential of trajectory based signatures, we propose a localization system, which consists of a centralized server and mobile clients, to perform the following tasks:

Training data collection: Clients voluntarily submit the training traces. The outdoor traces include both GPS coordinates and corresponding magnetic field measurement. Whenever GPS is unavailable (*e.g.*, urban canyons), we can still annotate our training traces with the correct trajectory label as long as we can obtain occasional GPS locks. The indoor traces include location coordinates and the associated magnetic or Wi-Fi measurements. We can

annotate the training trace with the trajectory label as long as we know at least two locations on the traces. Clients need not be always connected to the server. Instead, they only send data to the server when the Internet is available with low cost (*e.g.*, free Wi-Fi).

To simplify processing of testing traces, the training traces are cut into a few standard segments. For example, in outdoor traces each segment can be between two adjacent intersections. In our evaluation, we collect the training traces per segment, so they are already cut appropriately. For outdoor traces with GPS information, cut-points could be easily determined based on GPS. For ease of data collection indoors, we could leverage gyroscope and compass information [105], to detect common points that people make turns and use these points as intersections to cut the training trace. Additionally if a user has volunteered for crowd-sourced training data collection (*e.g.*, in return for coupons), she could facilitate cut-point detection (*e.g.*, by shaking the phone in a particular pattern which is recorded by the accelerometer). If a segment has multiple training traces, the server clusters them into one or multiple groups based on the similarities of the traces.

Localization: Localization is done locally without contacting the server to reduce communication cost and protect the users' location privacy. A mobile client downloads the training traces for the few areas it visits, occasionally (*e.g.*, once a week) when it is connected to Wi-Fi. Training traces are clustered as explained above to greatly reduce the amount of data to be downloaded by the client. If it happens to go to a new area, it can download new training

traces on demand. At the beginning of a trip, a mobile client first gets a GPS reading to locate the initial trajectory and then turns off the GPS. Then periodically the client may turn on GPS (*e.g.*, once every two minutes). Between two GPS readings, localization is performed by collecting the location signature and comparing the new trace with the training traces.

4.2.2 Problem Formulation

We decompose the localization problem into two steps: (i) identifying trajectory and (ii) localizing points on the trajectory. The first problem can be viewed as matching two time-series, where we are given measurements from a set of trajectories and our goal is to identify the trajectory whose measurement best matches our current measurement. Specifically, let m' denote our current measurement, and $m(i, j)$ denote a training trace for the i-th trajectory during j-th measurement, since a trajectory may be measured multiple times. We want to determine if the trajectory of the new trace appears in the training trace; if so, find $m(i, j)$ that best matches with m'. Once we find the correct trajectory, the second step is then to further localize the end point of the new trace, which is the user's current position and denoted as e, by aligning the current trace with the matching training trace and localizing e as the location of the point on the training trace aligned with e.

4.2.3 Trajectory Matching & Localization

4.2.3.1 Distance Metrics

A key issue in trajectory matching is to define an appropriate distance function that quantifies the similarity between the location signatures and is robust against different devices, speeds, time, environmental factors while capturing the general trend. Some simple distance functions include average and standard deviation of the time-series. However, such high-level aggregate statistics lose valuable details about the time-series.

Another natural distance function is the length of the longest common subsequence (LCSS) [25]. Given two sequences, it finds the longest subsequence common to them and uses the length to quantify their similarity. It allows skipping non-matching points. LCSS is well known and can be efficiently solved by dynamic programming. However, it has a few notable limitations. First, it uses a binary indicator for match or mismatch. We therefore need to determine an appropriate threshold for declaring a match between X_i and Y_j, which is hard. A too small threshold may miss many true matches, while a too big threshold may lead to many false matches. Second, LCSS only captures the length of the match and completely ignores the unmatched parts, which is also important.

We use dynamic time warping (DTW) [69]. It overcomes the limitations of LCSS by computing the fine-grained distance over all points of the two sequences. It stretches or compresses portions of the input sequence to improve the alignment and minimize the total distance. This is useful when we try to

match traces collected from different speeds. Below we first introduce the classic DTW, and describe the challenges of applying DTW in our context. Then we propose a range of techniques to address them.

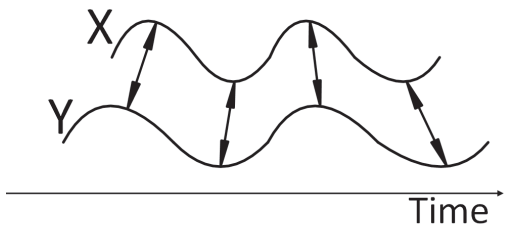


Figure 4.7: Alignment of two sequences in DTW

4.2.3.2 Classic DTW

DTW complete sequence: Given two sequences: $X = (x_1, x_2, ..., x_N)$ and $Y = (y_1, y_2, ..., y_M)$ and distance between any two points $d(x_i, y_j)$, DTW applies dynamic programming to find the best alignment of points on X to points on Y. The dynamic programming technique is general and can be applied to any additive distance function, and can be extended to multiple dimensions as well. We use L1 norm as the distance function in our evaluation. Other distance functions such as L2 norm yield similar results.

Figure 4.7 illustrates DTW alignment. DTW stretches required parts to find best alignment between the two curves. Algorithm 1 shows the pseudo code. It first initializes the distance between the two time-series to infinity, and then computes the DTW distance based on the recursive relationship in line 17.

DTW subsequence: Different from DTW complete sequence, DTW subse-

quence tries to find the alignment and the subsequence of Y such that it best matches with the complete sequence of X. DTW subsequence is more useful in our context because we need to localize a user while she is in the middle of a trajectory and the input sequence is not complete. In this case, DTW subsequence not only allows us to find a match, but also outputs the alignment which gives the current location of the user. It is interesting to note that DTW subsequence can be computed with the same complexity as DTW complete sequence, just by changing the initialization: line 13 to $DTW(1, j) = d(x[1], y[j])$ to allow first point in test to align anywhere rather than forcing it to align with the first point in the training trace. Furthermore, the DTW distance between the two sequences is $min(DTW(n, :))$ instead of $DTW(n, m)$ in order to find match for all elements in the test sequence to a portion of the training trace.

```
   Input: x, y: x – test and y – train, d(x[i],y[j]): distance between the two symbols.
   Output: DTW(x, y): DTW distance between x & y
 1 n = length(x);
 2 m = length(y);
 3 for i=1:n do
 4 |   for j=1:m do
 5 |   |   DTW(i, j) = infinity;
 6 |   end
 7 end
 8 DTW(1, 1) = d(x[1],y[1]);
 9 for i=2:n do
10 |   DTW(i, 1) = DTW(i-1, 1) + d(x[i], y[1]);
11 end
12 for j=2:m do
13 |   DTW(1, j) = DTW(1, j-1) + d(x[1], y[j]);
14 end
15 for i=2:n do
16 |   for j=2:m do
17 |   |   DTW(i, j) = d(x[i], y[j]) + min(DTW(i-1, j), DTW(i-1, j-1), DTW(i, j-1));
18 |   end
19 end
20 return DTW(n, m);
```

Algorithm 1: Compute DTW distance

4.2.3.3 Challenges of Applying DTW

Singularity problem: A well-known issue with DTW is that multiple points on one curve may be incorrectly mapped to one point on the other curve [51]. This is especially common when the Y-axis values of the two time-series are different. DTW is effective when aligning two sequences that are similar in Y-axis values except acceleration or deceleration in time. The algorithm faces difficulties when the two sequences differ in the Y-axis values as well. While global differences, such as global shifts or scaling or other linear transformation, can be removed before applying DTW, the local differences are much harder to deal with. For example, due to environmental noise and different measurement devices, the two time-series are likely to have different local features in the Y-axis values (*e.g.*, one series has a higher peak than the other). In this case, DTW responds to the difference in Y-axis values by modifying time-axis and causes incorrect alignment. Due to noise and external factors, our measurements from the same segment are unlikely to be identical and this may cause singularity problems and lead to inaccurate matching and localization.

Handling speed differences: When we find a test sequence in a training sequence, we require using the complete test sequence but can match to a smaller portion of the training sequence. If the test sequence is collected at a higher speed than the training sequence, ideally we want to match a point on the test sequence to multiple points in the training sequence. However, DTW fails to find such a match. Instead it prefers matching a point in the

test sequence to one point in the training sequence since the distance tends to increase with the number of matching points in the training trace.

Handling diverse training traces: Training traces for one trajectory can come from many users and they can be very different due to different device types, different time of the day, different positions of the devices in cars, and different cars, as well as random events and noises. We can improve accuracy by collecting more traces in more diverse environments. But how to effectively leverage different traces is an important design choice. Simply taking the average of DTW distance for each trace may mix up different patterns and lose important information. On the other hand, grouping traces based on how the measurement is collected may not be feasible due to lack of information. In addition, which factors in the measurement method affect the patterns may vary across different locations.

Minimizing computation cost and power consumption: Localization needs to be carried out on mobile devices, which have limited computation power and energy. DTW uses dynamic programming and involves building a large table: $M \times N$, where M and N are the sizes of the two input sequences. This can be computationally intensive. Continuous localization further requires DTW to be called more than once in one trajectory. In order to be a competitive alternative to GPS, the energy cost of computing DTW should be much lower than GPS. Therefore, we should not only achieve high localization accuracy but also low computation/energy cost.

Full path localization: Finally, continuous localization requires accurate identification of the completion of an old segment and the starting of a new segment. As a user is moving, the location signature is being collected continuously without clear separation of segments. While we can find the first segment using the initial GPS reading and then start localizing along that segment with the training traces, our algorithm needs to tell that the current segment is completed and quickly identify a new segment. To achieve that, a few important challenges are involved: first, it is not clear how to find the end point of each segment in a long testing trace. DTW subsequence can tell that the testing sequence is approaching the end of a training segment. However, it is hard to tell the exact end point due to data ambiguity and noise. The precision of the end point detection can significantly affect the accuracy of future segment matching. Second, given that the end point cannot be found precisely, it is important that our algorithm is robust to imprecise cuts.

4.2.3.4 Our Enhancements

Using wavelet coefficients to handle different speeds and achieve accuracy: Using wavelet coefficients instead of the raw traces offers two major benefits. First, it allows us to handle different speeds. When the two time-series are collected at different speeds, we use wavelet coefficients at different wavelet levels (or scales). In particular, as mentioned in Section 4.2.3.3, when finding X in a subsequence of Y, the classic DTW works well when X has lower or similar speed as Y; but the classic DTW subsequence fails when X

has higher speed than Y. Instead of matching X and Y wavelet coefficients at the same levels, we can find X in a subsequence of $Y(s)$, where $Y(s)$ is the Y's wavelet coefficient at level s. We use the level that gives the best match. We use lowest normalized DTW distance as the metric. It is defined as DTW distance divided by the length of the warping path to avoid bias towards shorter paths, where warping path refers to the set of matrix elements that define the mapping between X and Y. We consider both approximation coefficients and detail coefficients. Our results suggest that approximation coefficients yield higher accuracy.

Another important benefit of using wavelet coefficients is that it improves efficiency. If we use the approximation coefficients at one level higher, it reduces the number of coefficients by half, and decreases the computation time of DTW to $\frac{1}{4}$. Since the approximation coefficients at level 2 still give a good approximation to the original curve of observed magnetic field values, localization accuracy degrades little as shown in Section 4.3. Note that even if we need to match with wavelet coefficients from multiple levels, it is still efficient since the number of interesting levels is a handful. The longest wavelet coefficients we need to consider is only 1/4 of the length of the original time-series, which is 1/16 computation time. We use the Haar wavelets for our evaluation due to its simplicity and choose Level 2 as the default. For multilevel, we use level 2 for test trace, and pick the best matching train level between level 2 and level 5. Furthermore, if going from level 2 to level 3 does not improve DTW distance of the alignment, we stop trying higher levels.

This serves two important purposes: (i) improving efficiency, (ii) reducing the number of candidates and achieving higher matching accuracy.

Coping with singularity: We address the singularity problems by (i) preprocessing the data, (ii) using different local weights in DTW, and (iii) imposing a window constraint.

Specifically, to capture important features in the trace and reduce high frequency noise, we smooth the magnetic field traces using Savitzky–Golay filter [72]. We use more aggressive smoothing (with 45 points) for matching since only the high level shape is important. We smooth less aggressively (with 21 points) for localization since local features are more important. Furthermore, we observe that outdoor magnetic field traces from the same segment may look alike but with a shift in the y-axis. A shift is present sometimes even when using the same device. Therefore we remove the mean for these traces before alignment.

To favor horizontal, vertical, or diagonal direction in alignment, we can have local weighting as described in [69]. This is a simple modification to line 17 in Algorithm 1 to weigh $d(x[i], y[j])$ differently by w_h, w_v or w_d based on the step taken to reach (i, j). This gives the following recursion:

$$\begin{aligned} DTW(i,j) = min(&DTW(i-1,j) + d(x[i],y[j]) \times w_h, \\ &DTW(i-1,j-1) + d(x[i],y[j]) \times w_d, \\ &DTW(i,j-1) + d(x[i],y[j]) \times w_v) \end{aligned}$$

Having a lower weight on the diagonal (thus lower distance penalty) helps avoid singularity problem. Moreover, having a lower weight on the horizontal helps match a short testing sequence (which may be collected at a higher speed) in a longer train sequence. We use $w_d = 0.6$, $w_v = 1$, $w_h = 0.6$, which work reasonably well for all speeds.

Due to the singularity problem, some points in the beginning of the training trace may match with some point in the middle of the test trace. To prevent such an alignment from happening, we impose a *window* constraint as follows. Suppose the maximum ratio between the speeds of the two traces possible is max_speed_ratio. Then the n-th point in one of traces can only align with any point between $n \times \frac{1}{max\_speed\_ratio}$ and $n \times max\_speed\_ratio$ of the other trace. To accommodate this, we change the loops in Algorithm 1 as follows: (i) line 9 and 12: for loop goes from 2 to max_speed_ratio, and (ii) line 16: $i \times \frac{1}{max\_speed\_ratio}$ to $i \times max\_speed\_ratio$. Our implementation uses $max\_speed\_ratio = 2$ as the default.

Clustering training traces: Given that the time-series can exhibit different patterns for one trajectory, and it is challenging to identify the contributing factors of the different patterns, we use clustering to automatically partition training traces into different clusters as follows. We use the Meila-Shi spectral clustering algorithm [66], which is the recommended algorithm in [59] due to its excellent performance and solid mathematical foundation. Let W be the adjacency matrix of similarities between different pairs of training traces, with weight $w_{ij} = 1/DTW(i, j)$, where i, j are indices of training traces. Let

D be the degree matrix, which is a diagonal matrix with the node degree $d_i = \sum_j w_{ij}$ on the diagonal. We take the eigenvectors corresponding to the $\sqrt{N}$ smallest eigenvalues of the normalized graph Laplacian matrix $L_{\text{rw}} = I - D^{-1}W$ (where I is the identity matrix) and then call k-means clustering [57] to cluster points by their respective $\sqrt{N}$ components in these eigenvectors. We find the matching trajectory by finding the closest cluster, which gives the smallest average DTW distance to the testing trace. Then we find the closest training trace from this cluster and use this trace for localization within the trajectory. The server performs clustering on the training traces offline in advance.

Enhancing efficiency: Using wavelet coefficients already improves efficiency. To further reduce the computation cost, we observe that DTW is incremental in nature. Thus when DTW is called multiple times in one trajectory (*e.g.*, for continuous localization), we only compute the newly added rows in the dynamic programming table rather than computing the entire table every time. Specifically, every new reading corresponds to a new row, and the existing table entries can still be reused because of the dynamic programming property.

Full path localization: Putting together trajectory matching and point localization, we perform *Full Path Localization* of a user as follows.

1. Identify the current trajectory: DTW subsequence is run at every intersection to identify which segment the user takes.

2. Continuous localization within a trajectory: Once we know which segment the user takes, we periodically localize the user's current location to a location on the matching training trajectory using incremental DTW subsequence.

3. End point detection: When we find that we are close to the end of the training trajectory (*e.g.*, via DTW subsequence), we first call end point detection after waiting for a few seconds to ensure the trajectory is complete. Then our end point detection tries to find the training segment as a subsequence in the current testing trace using DTW subsequence. Assuming the testing trace covers a longer distance, the point on the testing trace that aligns with the end point of the training segment is considered as the road intersection on the testing trace.

4. Identify the new trajectory: After detecting the end point of the previous segment, we first wait and accumulate enough information about the current segment to allow correct matching. In our implementation, we wait for $min(min(L),\ fraction \times max(L))$ points, where L is the set of lengths of training traces. We use $fraction = 0.5$ outdoors and $fraction = 0.75$ indoors. A larger value is used indoors because segments are much shorter (*e.g.*, 4-6m vs. 100-200m). Then we apply the DTW subsequence in step 1.

   Since end point detection may have errors we not only use the most recent segment but also concatenate the previous segments to enhance

robustness while keeping the running time low. Since matches for the previous segments are known, our test candidate set does not grow. As before we use the 3 possible next segments for our candidate set but just concatenate them with the known previous segment, just to remove the dependence on end point detection. In particular, if the matching results based on the recent one segment and the recent two segments agree, we use the results. Otherwise, we use the last three segments for matching, and take a majority vote. In case there is no majority, we use the result based on the last three segments since it contains more information. In all cases, incremental DTW is used to improve efficiency.

4.3 Evaluation

4.3.1 Evaluation Methodology

Magnetic field: We use smartphones to collect magnetic field traces both indoors and outdoors at 32Hz, and also record the GPS readings for the ground truth outdoors. We use three different devices: Pantech Crossover, Huawei Prism, and Samsung Galaxy S. The recorded magnetic fields are in x, y and z directions with respect to the phone. We only use the magnitude since the direction is sensitive to the phone orientation. Our indoor magnetic traces are over 2 floors of an office building and outdoor traces are from different cities. Table 4.3 and 4.4 show our outdoor and indoor traces, respectively.

Wi-Fi: We collect Channel State Index (CSI) and RSS traces using the tool [40] released for Intel 5300 wireless cards. We refer to RSS as the average

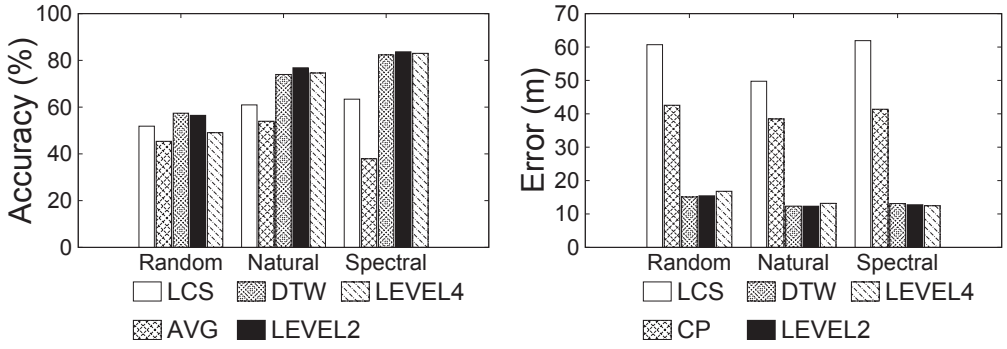
#	Area	Car	Device	Laps/Segs	Time
1	Shopping, Redmond	Accord (frontseat)	Pantech	3/10	Evening
2	Shopping, Redmond	Accord (frontseat)	Pantech	3/10	Night
3	Shopping, Redmond	Accord (backseat)	Pantech	3/10	Night
4	Shopping, Redmond	Accord (frontseat)	Prism	3/10	Evening
5	Shopping, Redmond	Fusion (frontseat)	Pantech	5/10	Night
6	Downtown, Redmond	Accord (frontseat)	Prism	5/17	Mid-day
7	Residential, Bellevue	Accord (frontseat)	Prism	5/14	Evening
8	Residential, Austin	Yaris (frontseat)	Prism	4/10	Night
9	Downtown, Austin	Yaris (frontseat)	Prism	4/10	Mid-day

Table 4.3: Table of outdoor magnetic field traces collected

#	Office Area	Device	Laps/Segs	Day & Time
1	Near cubes	Pantech	5/11	Weekend evening
2	Near cubes	Prism	5/11	Weekend evening
3	Corridors	Pantech	5/9	Weekday night
4	Corridors	Prism	5/9	Weekday night

Table 4.4: Table of indoor magnetic field traces collected

received signal strength over all subcarriers for that transmit-receive antenna pair. CSI is essentially fine-grained signal-to-noise (SNR) information for every OFDM subcarrier (or subcarrier group) for every transmit-receive antenna pair. We collect the Wi-Fi traces in an office building within an area of 225 m^2. We try to maintain uniform walking speed while collecting the traces. This can also be achieved using a Roomba robot as in [91]. We collect the traces for all possible paths given the floor layout and 3 traces for each path. They contain 11 segments. Our measurement uses MCS 0 [65] and transmission power of 15 dBm. Our evaluation uses magnitude of CSI. CSI is more fine-grained than RSS, so is expected to yield higher accuracy.



(a) Trajectory Matching (b) Localization Median Error

Figure 4.8: Outdoors: schemes across different clustering

4.3.2 Magnetic Field: Matching & Localization

4.3.2.1 Outdoors

Comparison of different schemes: Figure 4.8(a) compares the matching accuracy of DTW based approaches with longest common subsequence (LCS) and average (AVG) using traces 1-5. DTW is DTW on original time series, *i.e.*, y-value after the pre-processing mentioned in Section 4.2. LEVEL2 and LEVEL4 are DTW run on the approximate wavelet coefficients at level 2 and level 4. We omit the curves of other levels for visual clarity. For LCS, we use 10% of the range of the training curve as a threshold for a match. AVG uses the mean over an entire trajectory and finds the trajectory with closest mean. For all the schemes, we compare three clustering methods: random clustering, natural clustering based on the collection setting (*e.g.*, car type, device locations, etc.), and spectral clustering described in Section 4.2.3.4. We refer to different drives/walks of the same route/path as a *lap*, a lap may contain more than one segments. We first pick one lap as the test lap, and everything else is the training lap. Next, at every intersection, we consider

all possible test cases, where user reaches the intersection via any of the 4 intersecting segments and leaves the intersection via any of the remaining 3 segments. This gives us 12 test cases at every intersection for every test lap. We further vary the test lap one by one to get $12 * NumIntersections * NumLaps$ number of total test cases and present the aggregate results over these test cases.

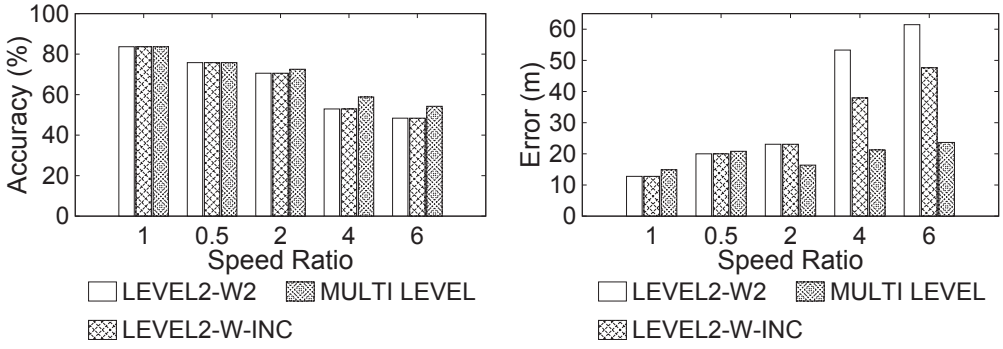
We make the following observations. First, DTW based approaches consistently achieve the highest matching accuracy. DTW accuracy is 10-30% higher than LCS and 25-117% higher than AVG. Second, using wavelet coefficients has comparable accuracy as original DTW. Third, spectral clustering gives better performance than natural clustering (by 10%), which outperforms random clustering (by 23%). It shows that traces collected from the same trajectories exhibit different patterns and clustering helps capture these patterns and improve the matching accuracy. Spectral clustering is better than natural clustering because the traces collected in the same way may not necessarily exhibit the same pattern. We use the traces 1-5 with spectral clustering as the default outdoor trace below. We also refer to it as *Combined* trace.

Figure 4.8(b) compares localization accuracy. On each trajectory we pick a point at 25%, 50%, and 75% of the total length and test for localization accuracy at each of these selected points. In LCS, we pick the last point returned on the training trace as the estimated current location. In CP, we match the current location in the testing trace to the closest matching signal value in the training trajectory. So CP is the only scheme that does not

use the trajectory information. Figure 4.8 (b) plots the median error. We compute error based on the GPS ground truth. We observe a similar trend as the matching results. DTW based approaches significantly outperform LCS and CP. For example, with spectral clustering, the median error of DTW (on the original time series) is around 13m, while the error of LCS is 62m and that of CP is 41m. The corresponding numbers for mean error are 16m, 63m and 48m, respectively. LEVEL2 performs as well as DTW on original time-series with the median and mean errors close to 13m and 16m, respectively. Both the matching results and localization results suggest using wavelet level 2 coefficients achieves similar accuracy as using the original time series. So we use LEVEL2 below due to its higher efficiency. We note that the mean and median errors show similar trends and henceforth only report the median error in the interest of space.

Impact of varying speed: Now we evaluate the robustness of our approach against different speeds. We manipulate the trace to simulate different speeds and show the result in Figure 4.9. Speed ratio 1 refers to the speed of the original non-modified traces although in reality they may not be collected at exactly same speeds. To simulate speed ratio of testing trace to training trace greater than 1 (*i.e.*, the testing trace is faster), we linearly interpolate the training trace. Similarly, to achieve speed ratio that is less than 1, we interpolate the testing trace. In this set of results, we use our MULTI LEVEL scheme as described in Section 4.2.3.4, with the base level of 2 and window size of 2. We omit DTW in the figure for visual clarity, since its trend is similar to

LEVEL2. We compare with LEVEL2 with window set to $max\_speed\_ratio = 2$ (LEVEL2-W2) and with window set to the speed ratio (LEVEL2-W-INC). The latter has the benefit of knowing the speed ratio, which is not available in practice. Despite the benefit, our MULTI LEVEL performs better than both versions of LEVEL2: its matching accuracy is 73% and 54% for speed ratios of 2 and 6, respectively; and its localization error is 16m and 24m, respectively. In comparison, the accuracy of fixed wavelet level drops significantly: the matching accuracy of LEVEL2-W2 and LEVEL2-W-INC drops from 84% to 71% under a speed ratio of 2 and drops to 48% under a speed ratio of 6; their localization error increases from 13m to 23m under a speed ratio of 2, and increases to 61m and 48m, respectively, under a speed ratio of 6. These results demonstrate the effectiveness of the multi-level wavelet scheme.



(a) Trajectory Matching (b) Localization Error

Figure 4.9: Outdoors: effect of varying speed

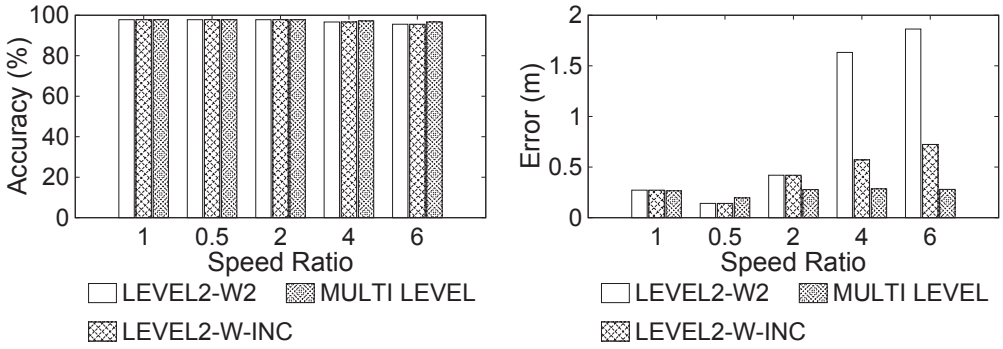
4.3.2.2 Indoors

Next we evaluate the performance indoors. For localization error calculation, since GPS is not available indoors, we set up an indoor coordinate

system by getting coordinates of the first and last points of the segment and walking at a uniform speed so that we can get intermediate locations through interpolation. For brevity, we only show the results for varying speed in Figure 4.10. The default speed ratio of 1 is also included for comparison. We consider traces $1-2$ together and $3-4$ together and use spectral clustering. First, we find that indoor accuracies are higher than outdoors due to more stable measurements and slower speeds. Second, matching accuracies do not deteriorate with higher speeds as indoor traces have mean information, which does not vary with the speed. Third, the MULTI LEVEL is robust to such speed variation, and outperforms LEVEL2-W2 by up to 83% and LEVEL2-W-INC by 57% at a speed ratio of 6. In comparison, the schemes based on the fixed wavelet level degrade significantly with the speed: the error increases from 0.3m to 1.8m for LEVEL2-W2, and from 0.3m to 0.7m for LEVEL2-W-INC between speed ratios of 1 and 6. Although here we show matching accuracy between 3 candidates, we find that even finding match among many more candidates indoors is accurate. This is useful when we do not have an idea of which intersection the user is at.

4.3.3 Magnetic Field: Full Path Localization

In this subsection, we study the accuracy and efficiency of our complete solution with magnetic field in realistic full path localization. The accuracy here is different from the previous evaluations because here the error is cumulative. Specifically if we make a wrong match on the current segment, all



(a) Trajectory Matching (b) Localization Error

Figure 4.10: Indoors: accuracy with varying speeds

subsequent matches will likely be wrong since it considers the end point of the current segment as the starting point of the new segment and uses that to determine the set of candidate training traces to match against. Similarly, the localization error also increases.

4.3.3.1 Outdoors

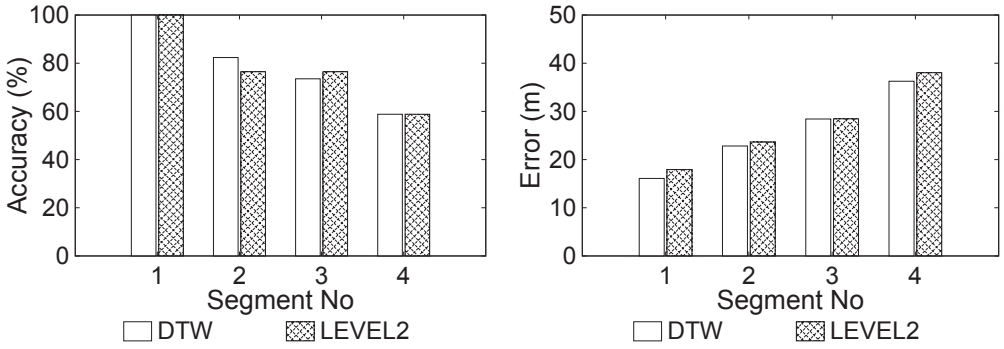
We show the benefit of our localization scheme in terms of (i) accuracy and (ii) power consumption.

Accuracy: We use the *Combined* trace, which has 3 intersections. At each intersection, there are 3 candidate next segments. We consider 2 routes: the route we drove on and the reverse route, and present the aggregate errors. Each route has 17 test laps, which gives 34 routes in total for testing.

In Figure 4.11, we show the performance at each segment in the path. When there is a matching error, we compute the future localization error until the current incorrect segment is over. Beyond that, we may not have traces for real error computation. Whenever there is no trace, we assume the estimated

location is the last point of the incorrect segment for error computation.

We assume the first segment is known (*e.g.*, from GPS lock) so the matching accuracy is 100%. As we would expect, the initial segments have highest matching accuracy and lowest localization error. As the user travels more segments, the accuracy degrades due to cumulative errors. Matching accuracy goes down from 75% at the second segment to about 60% at the fourth segment. Localization error is 18m on the first segment, and about 40m on the fourth segment. So we recommend getting a new GPS reading after the fourth segment.



(a) Trajectory Matching (b) Localization Error

Figure 4.11: Outdoors: accuracy across intersections

Power consumption & running time: Figure 5.16 shows our power benefit. We run computation on the phone since offloading the computation to a server requires communication and cellular communication is power intensive. We use power monitor from Monsoon Solutions [76] for power measurement. We measure the average power consumption when (i) phone is on, but screen is off and no application is running (p_1), (ii) phone is on, screen is on, and no other application is running (p_2), (iii) GPS is running (p_3), (iv) our localization

scheme is running (p_4). We implement our scheme as follows: we keep GPS on until we receive one GPS lock, then turn off GPS and only dump magnetic field readings. We use LEVEL2 in the implementation. We assume 4 segment drive time of 2 minutes, which is the case in our *Combined* outdoor trace. So each segment takes 30 seconds. We get sensor reading at 32Hz as in our evaluation and compute matching 18 seconds after the start of every segment (to accumulate enough readings) and afterwards localization is run every 6 seconds till the end of the segment. On Samsung Galaxy S3, average running times (over 30 minutes drive) for each localization and matching operation are 47 ms and 274 ms respectively. Thus it is very practical to run the computation on the phone. Even on the much older Huawei Prism, these numbers are 82 ms and 1883 ms respectively. After 2 minutes, we turn on the GPS (automatically) and repeat the matching and localization as explained above for next set of 4 segments and plot the benefit of our scheme for various drive times. We compute the power savings as follows: get power consumption of GPS as $p = p_3 - (p_2 - p_1)$ and our scheme as $p' = p_4 - (p_2 - p_1)$. The benefit of our scheme is then: $\frac{p-p'}{p}$. This includes CPU cost incurred due to localization. We see that the power saving varies across devices because different devices consume different power for the same computation and also have different GPS tail power. The power savings of our scheme can be as high as $45\% - 55\%$. Assuming 2000 mAh battery capacity on the Galaxy phone, the power saving corresponds to 48 minutes more battery life (2.5 vs. 3.3 hours) with the screen on, and 8 hours more battery life (6.4 vs. 14.4 hours) with the screen off.

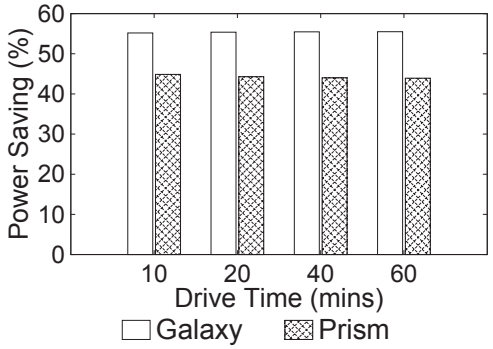
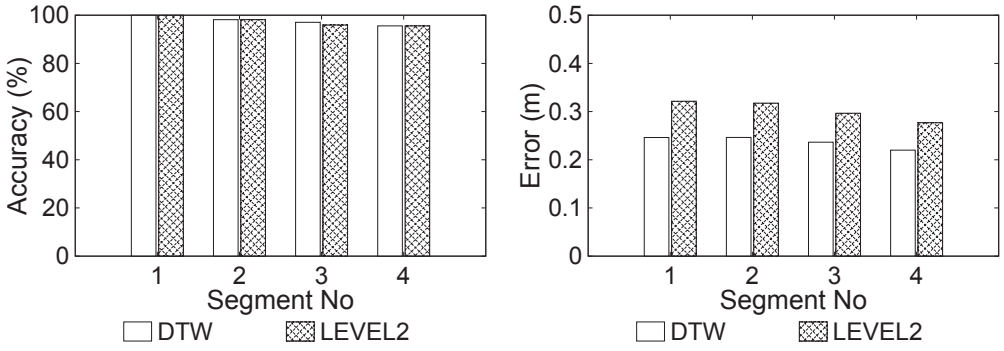


Figure 4.12: Power savings of our scheme

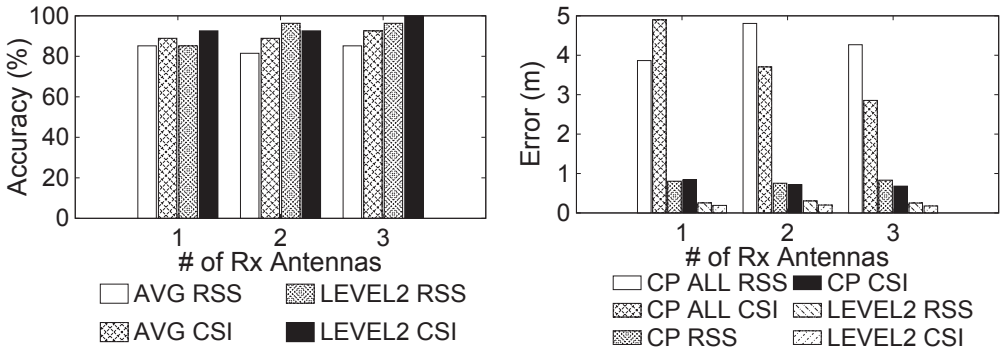
4.3.3.2 Indoors

Figure 4.13 compares full path indoor localization. Since it is hard to find multiple intersections with all 3 candidate options to take indoors, we collect nearby segments on an office floor, and create 3 intersections. We then create 27 different paths through these intersections. The results show a similar trend to the outdoor case, where performance degrades as the user moves along. Interestingly, the absolute performance numbers are much better indoors due to a much slower moving speed and more stable indoor environment. Even after 4 segments, the matching accuracy is still as high as 96% and the localization error is only 0.3m.



(a) Trajectory Matching (b) Localization Error

Figure 4.13: Indoors: accuracy across intersections



(a) Trajectory Matching (b) Localization Error

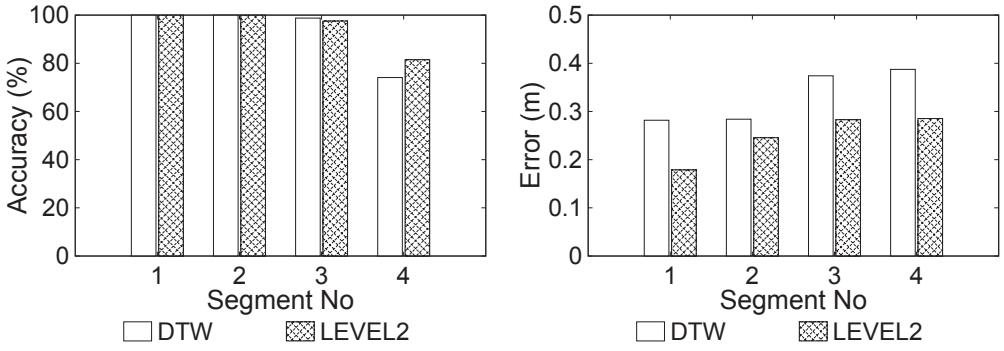
Figure 4.14: Wi-Fi: varying number of receive antennas

4.3.4 Wi-Fi: Matching & Localization

Next we use Wi-Fi signals as the location signature for indoor localization. We first study the performance in matching. We consider the following baseline approaches: AVG RSS/AVG CSI, which uses the mean RSS/CSI over an entire trajectory. Note that these schemes still use trajectory information and are expected to be better than point based schemes. We use one sender and we vary the number of antennas from 1 to 3. For all schemes, we use magnitude of the CSI. As shown in Figure 4.14 (a), all schemes work reasonably well (*e.g.*, over 80% accuracy) because trajectory is a fine-grained signature and allows unique path identification. Among them, we find CSI generally outperforms RSS. For example, LEVEL2 at 3 receive antennas achieves 100% matching accuracy with CSI and 96% with RSS. That is expected because CSI is a richer signature (30 values per antenna pair) as compared to RSS (1 value per antenna pair). Also LEVEL2 is more robust than AVG, and yields 93% accuracy with 3 receive antennas.

Figure 4.14 (b) compares localization accuracy of our DTW based approaches with the following baseline schemes: CP ALL RSS/CP ALL CSI: given the current point, pick the point that has closest RSS/CSI signature among *all* the possible trajectories; CP RSS/CP CSI: pick the point that has closest RSS/CSI signature on the current trajectory. Unlike the matching results, DTW based approaches perform much better than others in localization. Without the specific trajectory information, the localization error of CP is as high as 2.8m with CSI, and 4.2m with RSS with 3 receive antennas. With knowledge of trajectory, the error of CP decreases to 0.7m with CSI and 0.8m with RSS. In comparison, LEVEL2 achieves 0.17m localization error with CSI and 0.25m with RSS. This demonstrates the effectiveness of our approach.

4.3.5 Wi-Fi: Full Path Localization



(a) Trajectory Matching (b) Localization Error

Figure 4.15: Wi-Fi: accuracy across intersections

Next we perform full path localization indoors using Wi-Fi signals. We construct test routes similar to indoor full path localization using magnetic field signals. As shown in Figure 4.15, the accuracy of matching and localiza-

tion across different segments is high. Even after 4 segments, the matching accuracy is over 80% and localization error is only 0.28m.

Chapter 5

Location based Physical Analytics[1]

Physical analytics is an important application leveraging location information. This chapter presents our system ThirdEye that enables physical analytics in retail spaces. Section 5.1 describes our AutoLayout algorithm that fuses Wi-Fi, inertial sensor and camera data from glass users to simultaneously localize users and build product maps, Section 5.2 presents its evaluation in two large retail stores in the United States, Section 5.3 presents the algorithms and evaluation for classifying mobile user behavior into walk, dwell, gaze and reach-out, Section 5.4 presents attention identification within the field of view and finally Section 5.5 presents the energy measurements.

5.1 AutoLayout Design

As discussed in Chapter 1, an important requirement to enable physical browsing analytics is information regarding the layout of products within a store. However, given the immense number of stores in the world, including the many that a single shopper could visit over time, and the likely reluc-

[1]This chapter is revised version of the work in [81]. I was responsible overall for the project. Aishwarya helped with many of the evaluations. Dr. Chintalapudi, Dr. Padmanabhan and my adviser Prof. Qiu mentored and supervised the project.

tance of store owners to share information with their competitors, a crowd-sourcing based solution that can organically build layout maps without any help from the stores is attractive. Further, since we do not wish to discomfort the shoppers during their shopping, we also do not wish to seek any inputs from the shoppers. Therefore, we propose *AutoLayout*, which automatically infers layout based on video feed, inertial sensing data, and Wi-Fi measurements obtained from the shoppers equipped with smart glasses. If individual store owners make partial or complete layout information available to us (*e.g.*, distance between certain products), we can incorporate it into our AutoLayout framework to further enhance accuracy.

Inferring layout from crowd-sourced data is challenging:

- We have no control over shopper behavior, *i.e.*, no control over how they walk or where they dwell.
- There is no ground truth available from the shoppers since we do not seek any input from them.
- While there is much existing work on building 3-D layouts from image or video data, these assume availability of well-known landmarks, which are usually not available in stores. Moreover, they also require extensive imagery with overlapping coverage, which is problematic in our context because we have no control over where the shoppers go or which way they look and recording the video continuously incurs prohibitive energy cost.

The key idea in AutoLayout is that while different shoppers walk in

different ways within the store, Wi-Fi APs and positions of products in the store can serve as anchors to align these different walks. The anchors can be identified using both Wi-Fi RSS and images from the videos. AutoLayout simultaneously estimates the product locations and tracks the shoppers in a virtual 2-D coordinate system. Further, AutoLayout constructs a Wi-Fi propagation model (based on EZ [22]) so that it can track both smart glass users and smartphone users, albeit with reduced accuracy for smartphone users.

5.1.1 Data Collected by AutoLayout

AutoLayout continuously collects inertial sensor data, Wi-Fi data and opportunistically collects video feeds (whenever the shopper is gazing at products).

Inertial Data: As shoppers enter a store, in the background AutoLayout constantly keeps collecting inertial sensing data (accelerometer, compass), and continuously counts steps and tracks the direction θ, as obtained from the Google Glass, as the shopper walks around the store. Thus, for the i^{th} step that the k^{th} shopper takes, a direction value θ_k^i is recorded.

Video Data: Suppose that after the m^{th} step, the k^{th} shopper stops to gaze at or reach out for a product within the store, then as discussed in Section 5.4, the camera on the Google Glass is turned on and various products within view are identified by processing the images. Let I denote the set of all products in the store. Each product I_j is also associated with a list L_j of $< m, k >$ pairs

indicating that the k^{th} shopper was in the immediate vicinity of the product at this m^{th} step. Note that the same shopper may approach the same product repeatedly.

Wi-Fi Data: The glasses perform periodic Wi-Fi scans to record the Wi-Fi signal strengths. Suppose that, the l^{th} Wi-Fi access point AP_l was seen by the k^{th} shopper at the n^{th} step and the measured signal strength was $r^{l,k,n}$. For each Wi-Fi access point, a list W_l is maintained that stores the values, $r^{l,k,n}$.

5.1.2 Joint Layout Construction and Tracking

The unknowns that AutoLayout needs to estimate are as follows:

- x_k^i, 2-D location vector of the k^{th} shopper after his i^{th} step
- z^j, 2-D location vector of j^{th} product within the store
- $< P_l, y^l, \gamma_l >$, P_l is the transmit power of the Wi-Fi AP_l, y^l is its location, and γ_l is the path loss constant for the Log Distance Path Loss (LDPL) propagation model [22]. This is because, AutoLayout uses the LDPL model to predict the received signal strength at various locations from the APs.

The approach taken by AutoLayout is to minimize the objective function, which quantifies various errors, as shown below:

$$\begin{array}{lcl} J & = & J_X + J_Y + J_{AP} \\ J_X & = & \frac{1}{\sigma_X^2} \sum_i \sum_k \|x_k^{i+1} - x_k^i - \hat{e}_k^i\|^2 \\ \hat{e}_k^i & = & \left[\begin{array}{cc} \cos\theta_k^i & \sin\theta_k^i \end{array} \right]^T \\ J_Y & = & \frac{1}{\sigma_Y^2} \sum_{j, I_j \in I} \sum_{<m,k> \in L_j} \|x_k^m - z^j\|^2 \\ J_{AP} & = & \frac{1}{\sigma_{AP}^2} \sum_l \sum_{r^{l,k,n} \in W_l} \left\| \begin{array}{c} r^{l,k,n} - \\ rss(P_l, \gamma_l, y^l, x_k^n) \end{array} \right\| \\ rss(P, \gamma, y, x) & = & P - 10\gamma \log\left(\|y - x\|\right) \end{array} \tag{5.1}$$

In Eqn 5.1, the overall error objective function J comprises of three parts. First, J_X captures the constraint that shopper locations x_k^{i+1} and x_k^i coming from two consecutive steps of the same shopper must be separated by a single step and may be subject to σ_X (0.1 in our implementation) standard deviation of error. Second, J_Y captures the fact that all shopper locations from where a particular product was gazed at by the shopper must be very close to each other (we use a variance σ_Y, 1 step in our implementation, to account for the fact that the shoppers may not be exactly at the same place). This term essentially ties all the tracks from all the shopper together into the same coordinate system. Third, J_{AP} minimizes the difference between measured RSS and estimated RSS based on the parameters describing the LDPL model for all APs across all observations made by all shoppers. Since no global reference frame can be established, AutoLayout arbitrarily fixes any one position of a shopper as the origin. Our implementation simply chooses $x_0^0 = \mathbf{0}$.

5.1.3 The Optimization

Optimizing Eqn 5.1 is no easy task since the search involves a large number of unknowns and the objective is non-linear with multiple local minima. Consequently, we take a four-step approach to solve the optimization problem.

1. Obtain an initial estimate for shopper locations x_k^i and product locations z^j using the *BFSLoc* algorithm described shortly.

2. Optimize the partial objective function $J_{part} = J_X + J_Y$ to refine x_k^i and z^j using gradient descent method.

3. Obtain an initial guess for all the AP parameters $< P_l, y^l, \gamma_l >$ by minimizing J_{AP} using gradient descent method based on the existing values for x_k^i and z^j.

4. Starting with current values of the unknowns as the initial values, we iteratively optimize the overall objective function J as follows: first search x_k^i and z^j's that optimize J while fixing APs' parameters, and then search APs' parameters that optimize J while fixing x_k^i's and z^j's, and iterate. We iterate until the objective function improvement is within 10^{-5}.

In the interest of brevity, we do not describe gradient descent, since it is a well known optimization method. We describe BFSLoc, the first step in the AutoLayout procedure outlined above.

BFSLoc: The key idea in BFSLoc is to estimate locations (either product location or shopper location) using the shortest possible explanation. BFSLoc starts by constructing a graph where each shopper's location or product's location is associated with a node as depicted in Figure 5.1. Edges exist between nodes that represent consecutive steps of the same shopper (*i.e.*, x_k^i and x_k^{i+1}). Edges also exist between nodes that represent a product location z^j and a shopper location x_k^m when the product has been seen at that shopper location ($< m, k > \in L_j$).

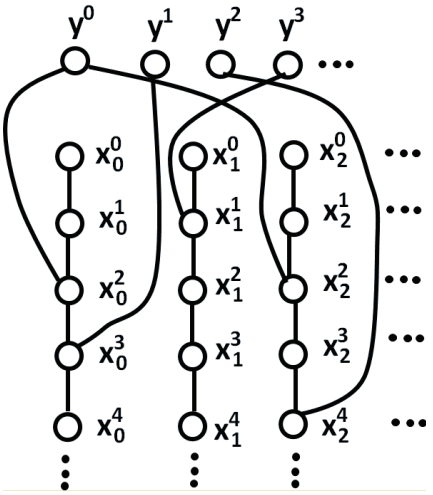
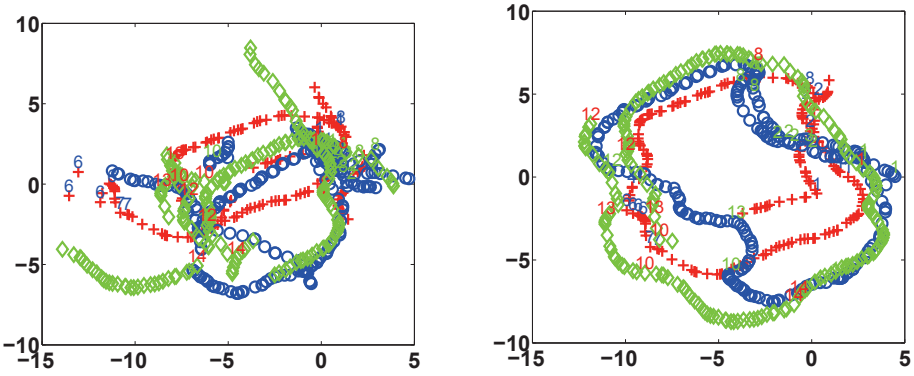


Figure 5.1: BFSLoc graph

The BFSLoc algorithm involves performing a breath first traversal on this graph starting from x_0^0 until all the nodes are traversed. Upon visiting any node a from a parent/previous node b, the value of the unknown corresponding to the node is computed as follows:

- if $a = x_k^{i+1}$ and $b = x_k^i$, compute $x_k^{i+1} = x_k^i + \hat{e}_k^i$.
- if $a = x_k^{i-1}$ and $b = x_k^i$, compute $x_k^{i-1} = x_k^i - \hat{e}_k^i$.
- if $a = x_k^i$ and $b = z^j$, compute $x_k^i = R(z^j, \rho)$, where $R(z, \rho)$ generates a random location in a disc of radius ρ with its center as the 2-D location z.
- if $a = z^j$ and $b = x_k^i$, compute $z^j = R(x_k^i, \rho)$.

The intuition here is that, starting from x_0^0 (which by definition is a certain location in our virtual coordinate space), we wish to estimate the location of all entities, whether a shopper at a particular step in their walk or a particular product, by following the fewest "links in the chain", with a view to minimize the accumulated error. We emphasize that BFSLoc is only the first step in AutoLayout procedure, so our goal in this first step is to obtain a good initial guess, not the correct estimates.



(a) After BFSLoc (b) After Optimization

Figure 5.2: Function example of AutoLayout

5.1.4 An Example of AutoLayout Functioning

In order to provide an intuition to the reader about the functioning of AutoLayout, we now show an example. In this example, three different shoppers were asked to take a rectangular route through the aisles in a retail store. The entire route was about 80m long. During the measurement, they occasionally walked, gazed and dwelled. Using the video, inertial and Wi-Fi

data from the three shoppers, we used AutoLayout to construct the product locations and the tracks of the shoppers. Figure 5.2 shows the layouts obtained after applying BFSLoc and after the entire optimization process. As seen from Figure 5.2 at the end of BFSLoc, the tracks of the three shoppers are very fragmented and do not overlap very well. This is due to the following reasons. First, there are errors in the compass measurements and in counting steps. Second, shopper stride lengths are different. However, after the optimization, AutoLayout not only determines the relative product locations but also identifies the entire path taken by the various shoppers. As a result in Figure 5.2 (b) the tracks after optimization look closer to the rectangular paths that the shoppers took. Moreover we also see AutoLayout inferred two large overlapping rectangles (where the two shoppers walked around the same set of aisles) and one smaller rectangle where the user turned around in the previous aisle.

5.1.5 Tracking Users in Real Time

For tracking shoppers in real-time, we use the same scheme as above except that we only consider the shopper's locations as unknown. Thus, the moment the shopper sees a product while gazing, a Wi-Fi scan is performed and his/her location is initialized. Thereafter, the subsequent locations are estimated by performing a gradient decent on J over only the shopper's locations. The optimization converges very rapidly in real-time since the initial location estimate is already quite close to the shopper's real location.

5.1.6 Dealing with Shopper's Head Movements

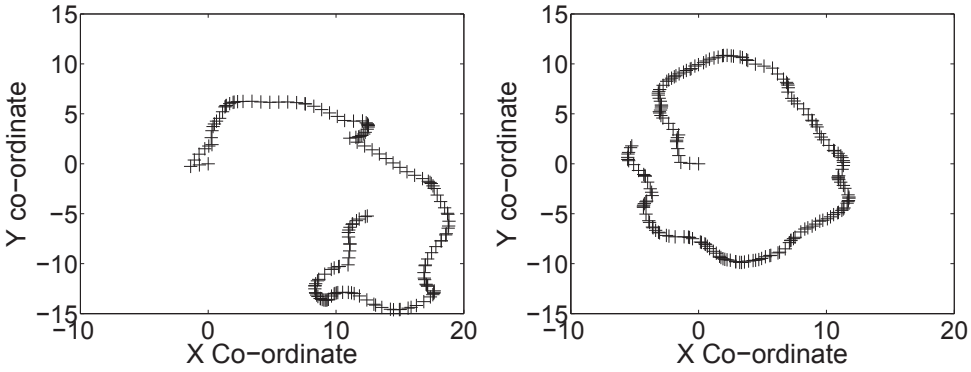
An important measurement in AutoLayout is the shopper's direction of motion. Since smart glasses are located on the top of the head and face the direction of the user, they typically are aligned in the direction of motion of the shopper. However, shoppers often turn their heads to look around and sometimes they walk with their heads oriented obliquely while watching something interesting or talking. Thus, a key challenge in using smart glasses is detecting the fact that the shopper has turned his head and not updating the direction of his/her path.

One obvious approach is to see if head movements can be detected and distinguished from actual change in direction of shopper's path. However, after studying several shoppers, we realized that it is often not possible to distinguish this based on accelerometer, compass or gyroscope measurements. Consequently, we take a very different approach based on the fact that most Google Glass users also carry their mobile phone along with them (usually in hand or pocket). The key idea is that when the shopper turns his/her head, the Google Glass compass orientation changes, but there is no change in the orientation seen by the mobile phone. Below we describe our scheme to correct for head movements.

1. We estimate the median offset between Google Glass and phone. As most of the time people walk while looking straight ahead, this accurately provides an estimate of the mobile phone's orientation with respect to

straight ahead as α.

2. Direction of user's heading is then, $\theta = \theta_{mob} + \alpha$ where θ_{mob} is orientation given by the mobile phone.

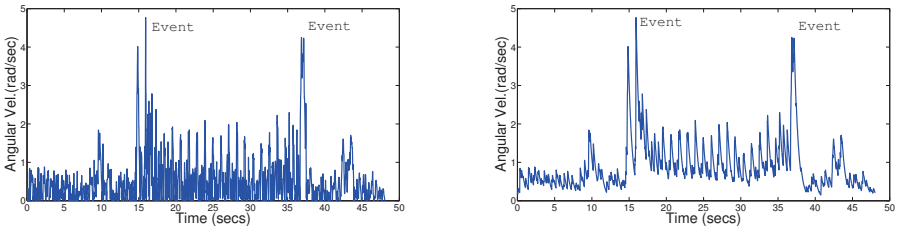


(a) Uncorrected (b) Corrected

Figure 5.3: Shopper tracks before and after compass correction

Figure 5.3 depicts the benefits of using the above scheme on a real shopper's track who started and came back to the entrance of the shop after taking about 100 steps (about 70m). This track was computed by simply adding $\hat{e}_k^i$ (the direction vectors). As seen from Figure 5.3, correction significantly improves the track quality.

Detecting when phone is taken out of pocket: One problem with relying on the shoppers' mobile phones for correcting the direction is that the shoppers might occasionally take their phone out to use it. However such events can be detected very reliably since they involve sudden motion. Once having detected such an event, we simply re-estimate the median offset from the Google Glass.



(a) Raw Signal (b) Onset Signal

Figure 5.4: Absolute value of angular velocity about Z axis

True detection [%]	False Alarm [%]
85	0
90	0.11
95	0.33
100	19

Table 5.1: Detecting change in phone position

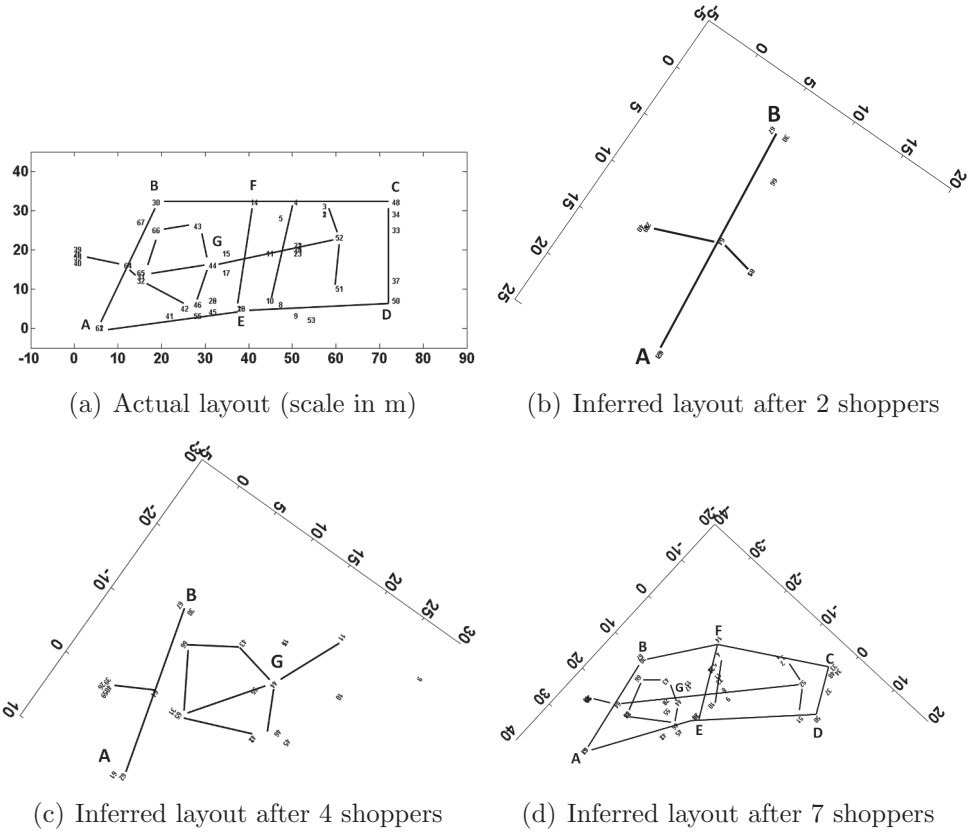
We show the events in the Figure 5.4 (a). We detect the onset of such events based on the absolute value of the signal as shown in Figure 5.4 (b). We examine 20 traces from 2 users, with each user taking the smartphone out of the pocket and putting it in the pocket for 5 of 10 of their walks. In the other 5 walks, no such event occurs. We compute the CDF of the onset signal over all the walks and vary the detection threshold from top 0.2% to 5% of the signal values. We apply this threshold to quantify true detection accuracy and false alarms. We quantify fraction of the times the event was detected versus the fraction of total seconds that we wrongly detected as pick up events. As shown in Table 5.1, we can detect up to 85% of phone putting in/pulling out of pocket events without any false alarms and up to 95% of the events with only false alarms during 0.33% of the total time of the walk.

We also tested other cases like holding the phone in hand and swaying hands while walking. We find that this action does not change the orientation significantly. For example, we found that the standard deviation of compass heading when user is walking with the phone held still in hands is 5.98 over 6 walks by 2 users where each walk was a straight line of about 25 m. It was 8.4 when the users were swaying their hands.

5.2 AutoLayout Evaluation

In this section, we evaluate two aspects of ThirdEye: the performance of AutoLayout and the performance of a shopping list reminder application for shoppers without smart glasses.

Experimental Methodology: We tested AutoLayout on 7 shoppers comprising three males and four females of different heights who shopped at a large grocery store (over 2 Km^2 in area) in the US. First, we performed a survey of the store, and measured the relative locations of about 67 different products for obtaining the ground truth. Each shopper was provided with a Google Glass and a smart phone that they would place in their pocket. The smart phone and the Google Glass communicated using Bluetooth. We shadowed each shopper as they went about their due course of shopping without interfering. Each shopper turned on Google Glass and smart phone at different locations within the shop (hence the start locations were different), visited different parts of the shop while taking different paths. While shadowing we noted the path that each shopper took and the locations where they gazed,



(a) Actual layout (scale in m)

(b) Inferred layout after 2 shoppers

(c) Inferred layout after 4 shoppers

(d) Inferred layout after 7 shoppers

Figure 5.5: Improvement of inferred layout with more shoppers for obtaining the ground truth.

Product Layout Generated by ThirdEye: AutoLayout generates a product layout of the store in a virtual 2-D coordinate system by merging the measurements from various shoppers with possibly different stride lengths. Consequently, the virtual coordinate will typically be a scaled, rotated (and possibly reflected), and sometimes distorted version of the original layout. However, one important property that it preserves is proximity relationships *i.e.*, products that are close physically will also be close in the virtual map.

First, to provide intuition into how the original layout compares to that inferred by AutoLayout, in Figure 5.5 we depict the original and inferred lay-

K	$\eta_{true}(K)$	$\eta_{false}(K)$
1	60%	40%
3	62%	29%
5	67%	26%
10	80%	18%

Table 5.2: Product layout proximity in AutoLayout

outs containing 67 different products within a large grocery store (a unique number is assigned to each brand in the Figure). To aid the readers in comparing the original and inferred layouts, we have connected lines between the locations of a few brands (labeled as A,B,C,D,E,F) that are located far away. Figure 5.5 depicts the evolution of the layouts inferred by ThirdEye as data from more and more shoppers is obtained. The evolution naturally depends on the exact areas where the shoppers visited. As seen from the Figure, after 7 shoppers visited the shop, the inferred layout approximately matches the original layout and proximity relationships are preserved in most cases (albeit rotation and scaling).

How well does AutoLayout Preserve Proximity Relationships?

In order to answer this question, for each i^{th} product we create the set $\mathbf{\Psi}_i(K)$ containing its K closest neighboring products using the ground truth data and the $\mathbf{\Omega}_i(K)$ containing its K closest neighbors based on the layout inferred from AutoLayout. We then evaluate average true detections $\eta_{true}(K)$ and average false detections η_{false} as

$$\eta_{true}(K) = \frac{\sum |\mathbf{\Psi}_i(K) \cap \mathbf{\Omega}_i(K)|}{\sum |\mathbf{\Psi}_i(K)|} \tag{5.2}$$

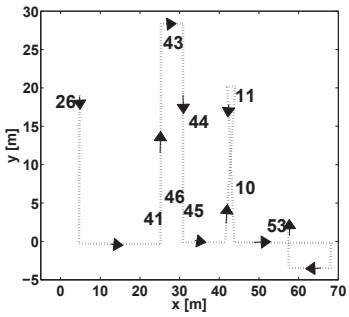
$$\eta_{false}(K) = \frac{\sum |\mathbf{\Omega}_i(K) - (\mathbf{\Psi}_i(K) \cap \mathbf{\Omega}_i(K))|}{\sum |\mathbf{\Psi}_i(K)|} \tag{5.3}$$

Here $|\bullet|$ represents cardinality of the set.

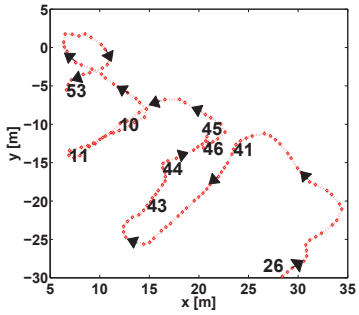
As seen from Table 5.2, AutoLayout predicts the closest product correctly about 60% of the time and reports incorrectly about 40% of the time. Note that predicting the closest product is particularly challenging given the close proximity of the products within the layout. The correct prediction ratio increases as the value of K increases and is about 80% for the set of closest 10 products. This can be very useful for a reminder application that can remind the shopper of a product they intended to buy when they are purchasing another product nearby or help provide a timely discount offer and convince them into purchasing another product that is close by.

How well does AutoLayout Track Shoppers With and Without Glasses?

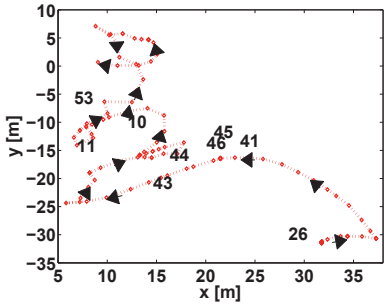
ThirdEye tracks shoppers in a rotated, scaled and potentially reflected coordinate system. Since it also constructs the product layout in this coordinate system, it can place the shopper relative to the products in the shop. Figure 5.6 (a) depicts the true path (ground truth) and Figure 5.6 (b) depicts the path inferred by ThirdEye in its coordinate system for one of the shoppers. In this case, the coordinate system is rotated, scaled and reflected. The products that the shopper passes by are marked by unique numbers assigned to them. In the example, the shopper starts at product 26 and walks to product 53, passing through a total of 9 products locations. As seen in Figure 5.6 (b),



(a) Original Path



(b) Path inferred by glass users



(c) Path inferred by smartphone users

Figure 5.6: How ThirdEye tracks shoppers' paths

ThirdEye successfully tracks the shopper in its own corresponding rotated, scaled and reflected coordinate system. Figure 5.6 (c) depicts the same path inferred by ThirdEye without using smart glass data, *i.e.*, using mobile-phone inertial and Wi-Fi data. As shown in Figure 5.6 (c), the inferred path is similar to the actual path though at a reduced accuracy.

How well does the reminder application work for shoppers without Glasses? We consider an application, where the shopper has a shopping list and is reminded to purchase as soon as he comes near one of the products in his shopping list. In order to evaluate how well the inferred map helps this application, we tested AutoLayout on shoppers without using their video from

K	$\eta_{true}(K)$	$\eta_{false}(K)$
1	17%	85%
3	29%	64%
5	38%	59%
10	60%	40%

Table 5.3: Performance for non-smart glass users

the Glass. We used AutoLayout to locate the person at each location (s)he dwelled (using only Wi-Fi and inertial data from his mobile phone) and then used the inferred map to determine all the top K nearest products and then compared with the ground truth layout map.

Table 5.3 depicts the true and false detection rates as we vary the number of items K. As seen from Table 5.3, errors are higher than the case of Glass users due to lower localization accuracy. Nevertheless with an increasing K, we are able to cover the top K nearest items more accurately, indicating not only that AutoLayout preserves proximity but even shoppers without smart glasses can benefit from the proximity information.

5.3 Behavior Classification

As discussed in Chapter 1, there are four modes exhibited by a typical shopper: purposeful walking, dwelling, gazing, and reaching out. We now describe these in greater detail and present the detection techniques used by ThirdEye.

5.3.1 Understanding the Behaviors

Upon entering a store, a shopper might walk purposefully towards a certain section of the store that they have in mind or in a certain direction. Upon reaching the section or aisle of interest, the shopper would typically slow down and dwell, i.e., amble around, say looking within an aisle for their product of interest. Occasionally, the shopper might stand at a spot and gaze at products on the shelf, visually scanning these to find the one of interest and perhaps making up their mind to reach out to it. Finally, the shoppers might reach out to the item of interest, either to pick it up or just to read the information on it.

The amount of time spent by the shoppers in each of the four modes would reflect their shopping intent. For example, shoppers who have time to kill or are unsure of what they intend to buy might spend a considerable length of time dwelling. Shoppers trying to compare and select among different brands/options of a specific item might be gazing. Finally, the act of reaching out indicates the greatest intent where the shoppers either intend to purchase the item or read information written on the item such as calorie contents or net weight. These behaviors may be exhibited in various sections of the store and may even be repeated by a shopper as (s)he comes back for a second look.

Figure 5.7 models the various behaviors as a finite state machine quantifying the probabilities of transitioning from one state to another at a granularity of one second. Table 5.4 reports the state transition probabilities estimated by studying the behavior of 7 shoppers who shopped in two large stores in the

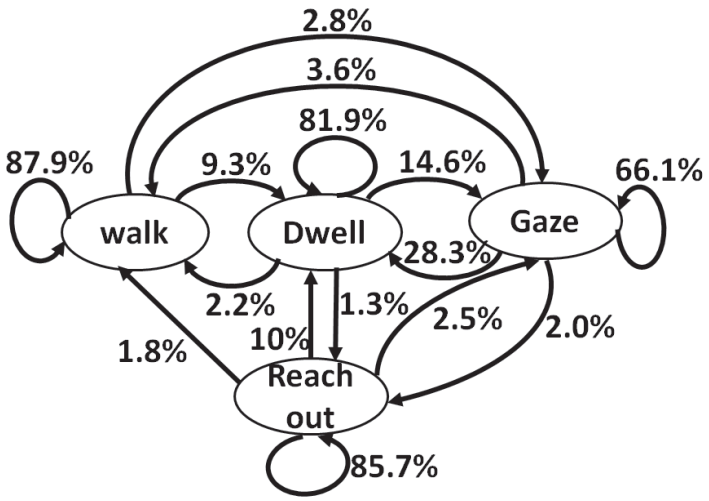


Figure 5.7: An example model for shopping behavior

	To					P(State)
	State	Walk	Dwell	Gaze	Reach Out	
From	Walk	87.9%	9.3%	2.8%	0%	17.3%
	Dwell	2.2%	81.9%	14.6%	1.3%	50.7%
	Gaze	3.6%	28.3%	66.1%	2.0%	23.8%
	Reach Out	1.8%	10%	2.5%	85.7%	8.2%

Table 5.4: State transition probabilities

United States, a grocery store and a department store, while wearing Google Glass. We shadowed the shoppers, taking notes as they shopped, interviewed them in the end, and finally studied the video obtained from their Google Glass. Thus, we manually classified the time spent by the shoppers in the store as walking, dwelling, gazing, or reaching out. We make the following observations from Figure 5.4 and Table 5.4.

- Shoppers tend to spend a majority of their time dwelling (50.7%) (as indicated by the P(State) column in Table 5.4), followed by gazing (23.7%) and walking (17.3%).

- Most frequent inter-state transitions are between dwell and gaze.
- For all four states, the high probability of remaining in the same state indicates that shoppers tend to continue in their current mode for several seconds before transitioning to another.

5.3.2 Behavior Classification Overview

ThirdEye automatically classifies shopper behavior into one of the four categories: walking, dwelling, gazing, and reaching out — using the inertial sensor data (*i.e.*, accelerometer and 3-axis compass) and the video feed from the camera of the Google Glass. ThirdEye also makes use of the inertial sensor data on shoppers' smartphones, where available, as discussed later in this section.

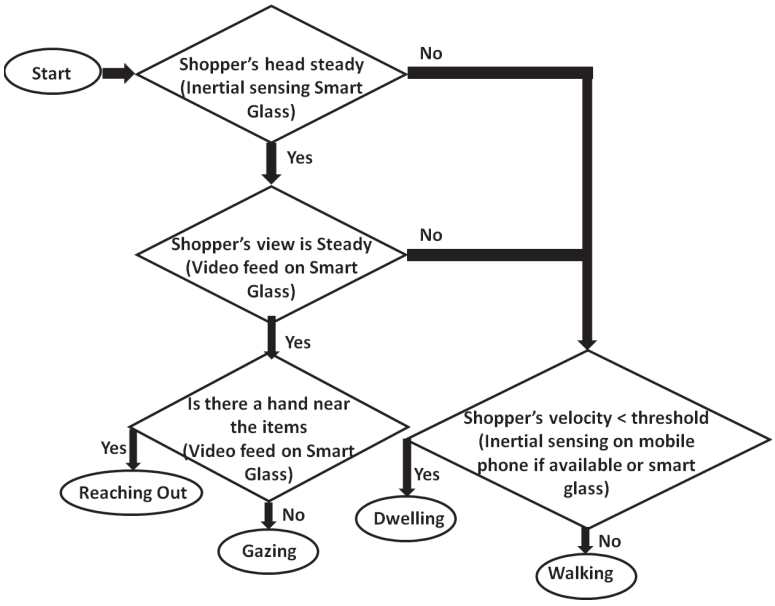


Figure 5.8: Overview of behavior classification algorithm

Turning on the video on Google Glass causes a high power drain on the device (as discussed in detail in Section 5.5), while inertial sensing results in much less power drain. Consequently, in our behavior classification scheme, we try to minimize the duration for which the video is turned on. Figure 5.8 depicts the high-level flow of our behavior classification scheme. We ran our scheme once every second, thus shopper behavior was classified at the granularity of each second.

Gaze detection: The first step in our scheme is to detect whether or not the person is gazing. Since shoppers typically hold their head steady while gazing and are also not walking, this can be detected by measuring the standard deviation in inertial sensor measurements on the smart glass. However, as will be discussed in Section 5.3.3, just relying on inertial sensors leads to a large number of false positives. Consequently, we use a preliminary detection based on a low standard deviation in the inertial sensor measurements on the smart glass as a trigger to turn on the video and analyze the scene to confirm that the person is indeed gazing (Section 5.3.3).

Dwelling and Walking: If there is a high standard deviation in the inertial sensors, this could mean that the person is either dwelling or walking. We use the step counter in Zee [79] to detect steps from the accelerometer. As discussed in Section 5.3.4, the accuracy of the step counter is higher when the accelerometer readings from the shopper's smartphone are used compared to when measurements from the smart glasses are used. This is because of spurious step detections caused by head movement. The step count is combined

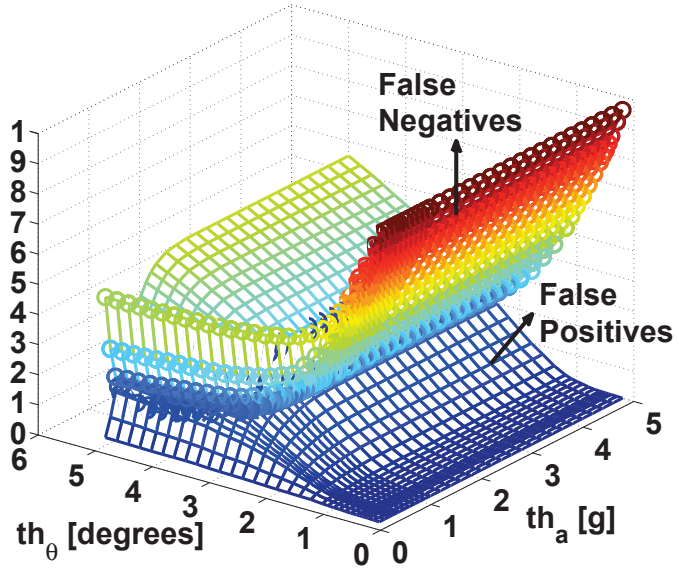


Figure 5.9: Choosing thresholds for gaze detection using inertial sensors

with compass data from the smart glasses to estimate the average velocity of the shopper. If the magnitude of average velocity is less than a certain threshold, the behavior is classified as dwelling, otherwise it is classified as walking.

Reaching out: In order to detect reaching out, we rely on detecting the shopper's hand in the field of view. As described in Section 5.3.6, we train an existing classifier [2] to detect hands in the video feed.

5.3.3 Gaze Detection

When a shopper gazes at one or more products, it reflects his or her interest in those products and so gaze is an important signal. We expect the person's head to be relatively steady while they are gazing. This can be detected using the inertial sensors on the smart glasses, in particular the

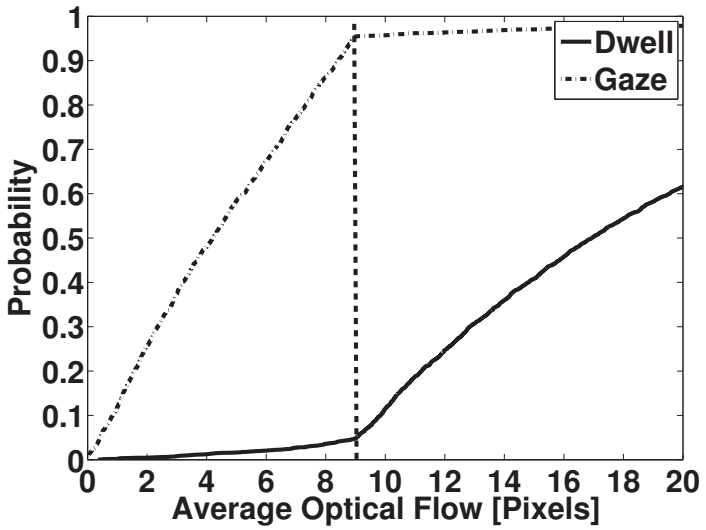


Figure 5.10: Choosing threshold for optical flow based gaze detection

accelerometer and the 3-axis compass. We use the rule $std(a_i) < th_a$ and $std(\theta_i) < th_\theta$ for $i \in \{x, y, z\}$, *i.e.*, the standard deviation in the acceleration values along all three axes must be lower than the threshold th_a, and the standard deviation along all three axes of compass must be lower than the threshold th_θ. Figure 5.9 depicts the dependence of false negatives (*i.e.*, missing gaze detection) and false positives (*i.e.*, mistaking other behavior as gazing) as a function of various combinations of threshold values. As seen from Figure 5.9, reducing th_a and th_{theta} decreases the false positives but increases the false negatives. While false negatives lead to missing actual gazing events, false positives will lead to turning on the video too frequently leading to a higher power consumption. In our implementation, we chose thresholds at $th_{theta} = 4.4$ degrees and $th_a = 0.7$g m/s^2, to limit false negatives to 10% while minimizing the false positives (at 33%).

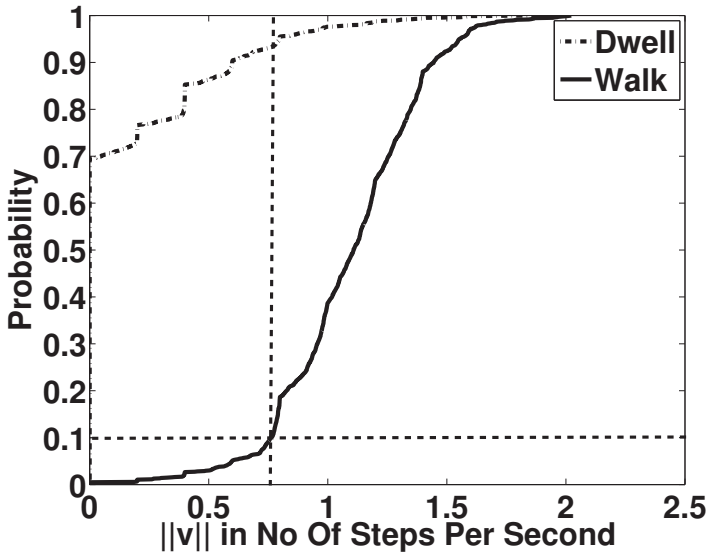


Figure 5.11: Choosing the threshold for dwell detection

Using optical flow to confirm gaze: However, 33% false positives is still too high. Since video has already been turned on now, one way to reduce this is to analyze the video feed to determine whether the person is indeed gazing. In order to achieve this, we rely on the property that when the shopper is gazing (s)he will intentionally make sure that there is very little change in what (s)he is seeing, from one moment to the next. Therefore, we used a vision based technique — *optical flow* [43] — applied to the video captured by the Glass. Optical flow measures the movement of pixels between consecutive images. We perform the optical flow computation on frames captured at 10 frames per second and calculate the average magnitude of the optical flow vectors over a one-second window. We then apply a threshold to this average optical flow value to decide whether the shopper is, in fact, gazing. Figure 5.10 depicts

the cumulative distribution function of the optical flow for dwell and gaze behaviors obtained from the labeled videos (from experiments where the camera was turned on all the time, to understand the difference between the optical flow values during dwell and gaze). As seen from Figure 5.10, a threshold of 9 pixels, achieves a high detection rate (97%) while keeping the false detections low (2%) if video is on all the time. We achieve a detection rate of 88% with 1.2% false detections for our energy efficient scheme where the video is turned on only after gaze is detected by the inertial sensors.

Finally, we shed some light on why gaze detection based just on the inertial sensors leads to a high false positive rate of 33%. We rule out the movement of a shopper's eyes while their head is held steady as a significant cause since the optical flow method, which also does no eye tracking, yields a low false positive rate of 2%. Based on a closer examination, we identify three reasons for the significant difference in the detection rate in the two cases: (1) the shopper stands at a spot and leans forward, *i.e.*, zooms in, while holding their head steady to examine a product closely, (2) the shopper is looking at an object that they are holding and turning around from side to side (*e.g.*, to examine the front and back of a product), and (3) the shopper is looking steadily at a distant scene, which is a more common reason. In each case, inertial sensors based detection concludes that the shopper is gazing while the optical flow based approach concludes the opposite. One could argue that at least cases (1) and (2) correspond to the shopper, in fact, showing attention in a manner that the notion of gazing is intended to capture. However, we

Device	Step Count Error [%]
Phone	2.6%
Glass	10.2%

Table 5.5: Step detection accuracy in mobile phone vs smart glasses

have taken the conservative position of restricting gazing to only cases where the visual image seen by the user is steady.

5.3.4 Dwell Detection

Dwelling refers to the shopper lingering around the same location. While the shopper could be taking steps during dwelling, they would tend to be ambling around, but roughly remaining in the vicinity, *i.e.*, having a small net displacement. Consequently, there could be significant movement during dwelling, so dwell detection based on simple approaches such as measuring the accelerometer magnitude or even looking for a reduction in the step rate, may not work correctly. Therefore, to build an effective detector, we rely on the intuition that unlike purposeful walking, in dwelling, the net displacement of the shopper will be very small, even though he/she may be walking around. In other words, dwelling is detected by a sharp drop in the magnitude of the average velocity vector of the shopper.

As discussed in Chapter 1, we use the step counter in Zee [79] to detect walking. Since most shoppers are likely to be carrying a smartphone with them, even if they are wearing smart glasses, measurements from either the smartphone or the smart glasses (or both) can be used to detect and count

steps reliably. In our experiments, we found detection to be superior when done in the mobile phone as compared to smart glasses, as indicated in Table 5.5. This is because a shopper's head movements during shopping are mistaken for steps. Consequently, in ThirdEye we use inertial sensors from the phone for step counting, whenever possible.

In order to detect dwelling, we continuously compute the net displacement over a window of the past τ seconds by using the compass in the Google Glass to provide the instantaneous direction of the shopper's motion. Suppose that the shopper takes K steps during a τ-second window. Further, suppose that at the i^{th} step, his/her orientation was θ_i. Then, we compute the magnitude of the net velocity vector as,

$$\|v\| = \sqrt{\left(\sum_{i=1}^{i=K} \cos\theta_i\right)^2 + \left(\sum_{i=1}^{i=K} \sin\theta_i\right)^2} \tag{5.4}$$

When $\|v\|$ drops below a certain threshold $\|v\|_{dwell}$ steps/sec, we conclude that the user is dwelling. In our implementation we set $\tau = 5$ seconds, under the assumption that any period shorter than 5 seconds cannot be considered as dwelling.

In order to choose the threshold $\|v\|_{dwell}$, we considered the distributions of velocities seen during walking and during dwelling, namely $\phi_{walk}(\|v\|)$ and $\phi_{dwell}(\|v\|)$ (depicted in Figure 5.11), as derived from the ground truth obtained from the videos collected from the shoppers. As seen from Figure 5.11, the threshold of 0.71 was chosen to allow 10% false errors (*i.e.*, walking is mistaken as dwelling) while providing a detection rate of about 95% (*i.e.*, dwelling is mistaken as walking 5% of the time).

Predicted⇒ Actual⇓	Walk	Dwell	Gaze
Walk	**90.4**%	9.1%	0.5%
Dwell	3.4%	**95.8**%	0.8%
Gaze	1.4%	10.7%	**87.9**%

Table 5.6: The confusion matrix for classification of walk vs. dwell vs. gaze

5.3.5 Summary of Dwell, Walk, Gaze Detection

Overall, using inertial sensing based dwell detection and inertial sensing plus optical flow based gaze detection, ThirdEye is able to effectively classify the shopper's traversal into walking, dwelling, and gazing. Table 5.6 shows the confusion matrix for this classification based on 3 hours of store visit data from 7 shoppers. The large values along the diagonal (in bold) show that the classification is correct in the overwhelming majority of cases. It is also worth remembering that there is an element of subjectivity in the ground truth, and that might account for some of the misclassifications.

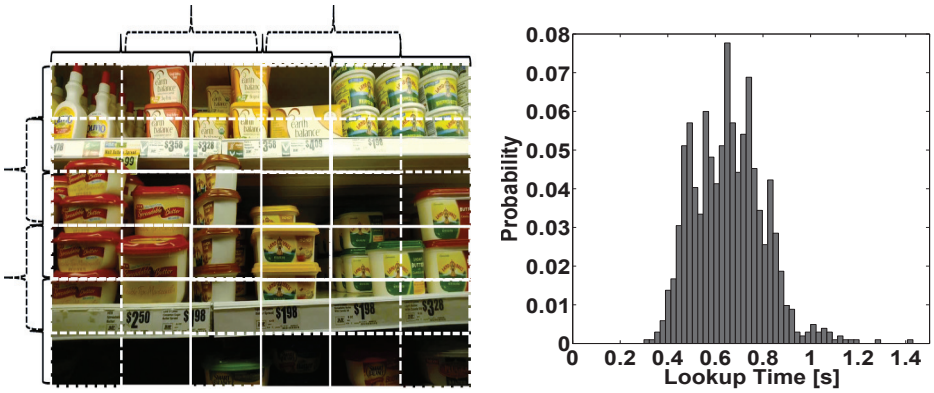
5.3.6 Reaching Out Detection

When the shopper eventually reaches out for a product, it indicates a very high degree of interest in that product. How can we detect such reaching out events? Unsurprisingly, we find that measurements from smart glasses or smartphone based inertial sensors are inadequate for distinguishing reaching out from gazing, since these sensors do not detect movement of the shopper's arm or hand.

Our detection scheme relies on the observation that in most cases of

(a) Two pieces of same hand (b) Spurious hand

Figure 5.12: Reaching-out detection



(a) Reverse image search example (b) Distribution of reverse search times

Figure 5.13: Reverse image search

reaching out, the user's hand becomes visible in the image, whereas it tends not to be seen in the image when the shopper is walking, dwelling, or gazing. This is because users tend to look at whatever they are reaching out for and moreover the camera in the smart glasses typically sees what the user sees.

To detect hands, we trained a TextonBoost classifier [2] on 100 hand images from various shoppers. The classifier identifies the areas covered by a hand and marks them as depicted in Figure 5.12. These figures also show two typical causes of error that arise from using the classifier, namely *hand fragmentation* and *spurious detections*. As seen in Figure 5.12 (a), the presence of a watch on the hand fragments the hand into two separate parts. On the other hand, in Figure 5.12 (b) the presence of items with similar texture and color as the hand results in spurious detections. To deal with fragmentation, we cluster together all fragments that either have overlapping bounding boxes or bounding boxes that are within 50 pixels of each other at their closest points. To deal with spurious detections, we rely on the observation that most spurious fragments tend to be much smaller in area compared to a real hand. Consequently we eliminated all fragments that occupied fewer than 1500 pixels (0.16% of the image). Overall, our reaching out detection scheme has a success rate of 86%, and a false detection rate of about 15%, based on testing over several videos from the training set. Further, we may miss detections when shoppers are reaching out for items in the lower shelves, as their hands may not appear within the field of view. Our reaching-out detection is currently offline since it involves more expensive vision processing and is an analytics step to understand the user's interests. Perhaps tapping sensor data from other devices such as smart watches would improve the reliability and computation cost of reaching out detection in such cases. Also, the use of more advanced pattern recognition techniques such as deep learning [52] could increase the

success rate while reducing the false detections. We defer an exploration of these directions to future work.

5.4 Attention Identification

Once it is established that the shopper is either gazing or reaching out, the next step is to identify the item(s) that the shopper is bestowing their attention on. Here, visual input plays a key role.

5.4.1 Reverse Image Search

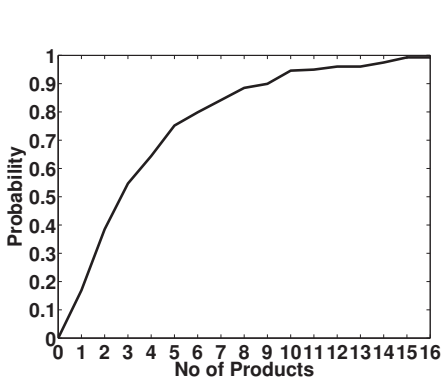
ThirdEye captures images and invokes a cloud service to perform reverse image search, *i.e.*, identify the objects contained within the image. In our current implementation, we use Google's service [3], although ThirdEye is not tied to it and could also invoke alternative services, such as Bing Vision [1] and Nokia Point & Find [4].

Reverse image search involves object recognition, which in turn involves extracting image features (*e.g.*, using an algorithm such as SIFT [58]), comparing these with precomputed features from a large corpus of existing images to find a match, and retrieving meta-data associated with the matching image(s) in the corpus (*e.g.*, product name). The scale and complexity of this computation necessitates running it on the cloud (*e.g.*, the corpus of product images could run into the tens of millions). In turn, this calls for being selective in how frequently reverse image search requests are sent to the cloud.

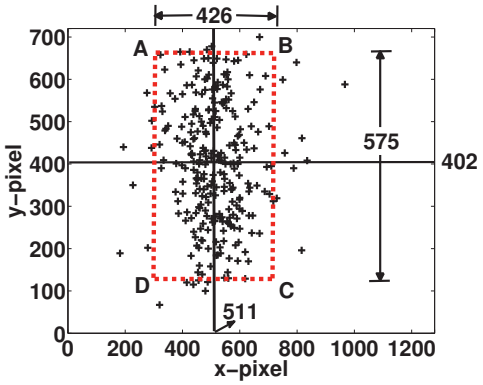
Therefore, only when gazing or reaching out are detected, does Third-

Eye capture images and invoke the cloud service. During each period of gazing, ThirdEye captures all the images during the gazing period. Each image is then divided into 25 overlapping parts (in order to avoid images being broken at the borders) and each part is sent up to the cloud separately to minimize the chances of object recognition failing because of image-complexity issues. An example is depicted in Figure 5.13 (a) with the overlapping parts that are sent for look up. In the example in Figure 5.13 (a), 2 (Lands o lakes butter and Earth Balance) out of 5 brands were detected by Google reverse image search. 3 were missed: Spreadable butter, Olivio, Smart Balance. Our evaluation based on manually studying the frames indicates that this scheme detects 53% of the products in the frame correctly. 7% are false positives, in other words, products are recognized incorrectly when they are absent. Finally, almost 40% of the products are never recognized. Further analysis indicated that almost all these 40% were not popular brand names but local brand names or items without labels like fruits and vegetables. Note that human computation (*e.g.*, through Amazon Mechanical Turk) could be used for brands that are not listed as a part of Google reverse image search and thus improve the database.

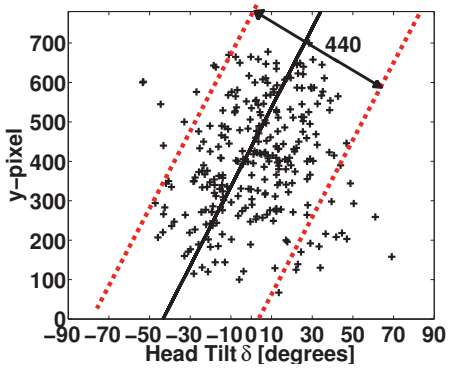
Figure 5.13 (b) depicts the distribution of reverse image look up times over all our look ups. As seen from Figure 5.13 (b), the look up times vary between 250 ms and 1 sec. with a mean of 650 ms.



(a) CDF of number of items seen in the field of view



(b) Distribution of center of attention



(c) Dependence of center of attention on tilt

Figure 5.14: Attention

5.4.2 Focus Within Field of View

In general, the number of items within the field of view of the Glass could be large. Figure 5.14 (a) depicts the cumulative distribution function of the number of items recognized in any given frame. As seen from the figure, in some cases the number of items seen can be as large as 16. It is likely, however, that the shopper is only focused on one or perhaps a small number out of these items in his/her field of view. Our goal, therefore, is to identify

the shopper's focus within the field of view.

It might be tempting to assume that the shopper's focus lies at the center of the field of view of their Glass. Figure 5.14 (b) depicts the distribution of center of attention from the videos from several shoppers as they gazed. To learn the center of attention, we shadowed each shopper (*i.e.*, walked with them) as they shopped and also interviewed them afterwards while playing back the video recording from their Glass. As seen from the figure, the center of attention of the shopper is not necessarily at the center of the screen but is distributed over various pixels in the view. The dotted rectangle ABCD in Figure 5.14 (b) depicts the area that includes 90% of the centers of attention as indicated by the shoppers. Based on Figure 5.14 (b), the region of the video that might include the center of attention with 90% probability is only about 27% ($\frac{575\times426}{1280\times720}$) of the frame.

Further, as seen from the figure, there is a clear bias towards the left of the field of view. In fact, the mean x value in Figure 5.14 (b) is $x_o = 511$. This is simply a consequence of the fact that the front-facing camera in Glass is mounted to one side (to the top-right of the right eye), so the center of the shopper's field of view lies to the left of that of the Glass.

Dependence of Attention on Head-Tilt: At a first glance, Figure 5.14 (b) seems to indicate that the variation along y-axis of the center of attention is random and has a spread across the entire view (580 out of 720 pixels). However, deeper analysis revealed an interesting fact – *the center of attention in fact depends on the tilt of the head of the person.* Figure 5.14 (c) depicts

the dependence of the y-coordinate of the center of attention as a function of tilt of the head δ, as measured by the compass on the Google Glass. As seen from Figure 5.14 (c), there is a mild linear dependence of the center of attention on the head tilt. We used RANSAC [36] (a popular scheme used in computer vision that is robust to outliers) to find the equation of the line to be $y = m\delta + c$, where $m = 10.05$ and $c = 436.38$ (depicted in Figure 5.14 (c)).

In hindsight, this makes sense, since when a shopper looks up or down towards a product, he/she accomplishes this by partly turning head up/down and partly by rolling their eye-balls. Consequently, the shopper's focus within their field of view would tend to shift in the same direction as their head is tilting. The region that includes the 90% of the centers of attention is also indicated by the dotted lines in Figure 5.14 (c). As seen from the figure, the spread around the line now has reduced to 440 (60% of 720) instead of 575 (80% of 720) pixels (in Figure 5.14 (b)).

Probability Distributions for center of attention: Since it is in principle impossible to determine exactly which item the shopper is focused on, we resort to assigning a probability value $P(x, y|\delta)$ to an item in his field of view at the x and y pixel position, given the head-tilt, δ, as obtained from the compass:

$$P(x, y|\delta) = \Phi_X(x - x_o)\Phi_Y(y - m\delta - c) \tag{5.5}$$

The distributions $\Phi_X(z)$ and $\Phi_Y(z)$ are obtained using histograms from actual data. Figure 5.15 depicts the function $P(x, y|\delta)$ as a function of $x - x_o$, and $y - m\delta - c$.

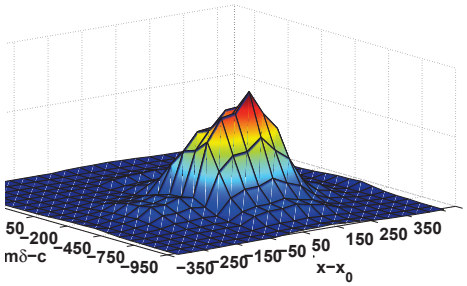


Figure 5.15: $\Phi(x, y|\delta)$

Scheme	Top 1	Top 2	Top 3
Naive	66%	80%	86%
Max-Likelihood	76%	86%	90%

Table 5.7: Determination of focus of attention

Evaluation of Focus of Attention: We now ask the question how accurately we can identify the item that the person is looking at during gaze? We consider two schemes to measure the relative importance of various items in view. Our first scheme, deemed Naive in Table 5.7, naively uses distance from the center of the video (*i.e.*, pixel 640, 360) while our second scheme, deemed Max-Likelihood in Table 5.7, uses $P(x, y|\delta)$. In Table 5.7 we measure the fraction of times the top ranked item corresponded to the correct item the shopper was gazing at and also the fraction of times where the correct item was among the top three ranked item.

As seen from Table 5.7 the Max-likelihood approach is 76% accurate in terms of pin-pointing the correct item being gazed at, while the Naive approach only succeeds 66% of the time. Further, about 90% of the time, the correct item is among the top three prospective candidate items.

Devices such as the Tobii glasses [5] perform eye tracking, which can

enable more accurate user attention identification than we present in this work, but these are sophisticated devices which include multiple eye cameras in addition to a front-facing scene camera.

5.5 Energy Measurements

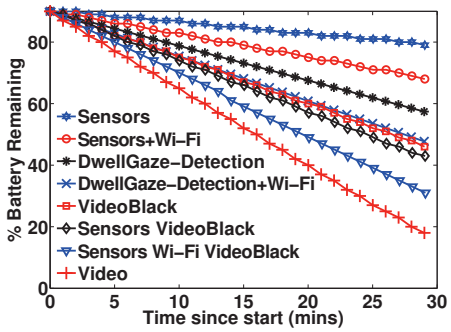


Figure 5.16: Power measurement on the Google Glass

In this section, we evaluate the power consumption of running the inertial sensors (accelerometer and compass), Wi-Fi, Video capture and our dwell and gaze algorithms on the Google Glass. Since we can not remove the battery and bypass it using a power monitor, we use the android API to monitor battery level [12] and log the remaining battery level every minute. We plot the result for various combinations as shown in Figure 5.16. We plot the battery level starting from 90% for the next 30 minutes. The legend and the description below are arranged in an increasing rate of power consumption for ease of reading.

Sensors: Here accelerometer and compass were sampled at the fastest allowed rate, *i.e.*, 200 Hz samples were received.

Sensors+Wi-Fi: Since AutoLayout uses Wi-Fi (to enable ThirdEye for non-smart glass users), here we measure the impact of performing back-to-back Wi-Fi scans while simultaneously sensing from accelerometer and compass at 200Hz.

DwellGaze: Here we evaluate the power consumption as the shoppers go shopping during their due course without Wi-Fi scanning. VideoBlack [102] is an application that takes video without displaying on screen so it is energy efficient compared to Video. In DwellGaze, VideoBlack is turned on only when the glasses detect that the person is gazing (this includes the effect of 33% false positives and 10% false negatives).

DwellGaze+Wi-Fi: Here we compute the same as DwellGaze except with back-to-back Wi-Fi scans for AutoLayout.

VideoBlack: Video is taken with screen off and is simply stored, no other computation is performed during the measurement.

Sensors+VideoBlack: During gaze, once video is turned on, we measure the impact of storing video while simultaneously sensing accelerometer and compass at 200Hz.

Sensors+VideoBlack+Wi-Fi: Here we measure the impact of running Wi-Fi scans back-to-back simultaneously with sensing at 200Hz and storing video using VideoBlack.

Video: Here the video on the smart glasses is turned on and stored. The screen is on while recording the video.

Scheme	Predicted Lifetime [m]
Sensors	265
Sensors+Wi-Fi	134
DwellGaze	89
DwellGaze+Wi-Fi	69
VideoBlack	75
Sensors+VideoBlack	70
Sensors+VideoBlack+Wi-Fi	49
Video	47

Table 5.8: Predicted lifetime

Table 5.8 depicts the predicted lifetime of various schemes based on our measurements. As seen from Table 5.8, VideoBlack provides almost 60% higher energy savings than using Video. Further, even VideoBlack is more than 3× expensive compared to using inertial sensors indicating the need for judiciously turning on Video. Continuous Wi-Fi also has a significant impact and typically cuts lifetime significantly. Optimized schemes can be designed to use Wi-Fi sparingly for AutoLayout, however we defer it to the future. Thus, the lifetime of ThirdEye is approximately between 70 min (for Wi-Fi turned on continuously) and 90 min (with Wi-Fi scanning turned off).

Chapter 6

Conclusion and Future Work

In this work we focus on designing accurate localization schemes for different mobile scenarios and building applications that leverage this information.

6.1 Dissertation Summary

First we consider a multi-hop mobile network, analyze both real and synthetic mobility traces and show that they all exhibit temporal stability and low-rank structure. Motivated by these observations, we develop novel localization schemes that exploit these characteristics while leveraging network topology information during each time interval. Using extensive simulation based on real and synthetic mobility traces and testbed experiments, we show that our localization schemes significantly out-perform the existing schemes.

Second we consider single mobile nodes and propose a novel localization scheme that uses measurements from a trajectory as a location signature. We identify the trajectory using our enhanced DTW subsequence, and localize points on the trajectory based on the DTW alignment result. To enhance the accuracy, robustness, and efficiency of DTW alignment, we perform multi-level

wavelet decomposition, cluster the training traces from the same trajectory, and enable continuous full path localization by combining trajectory matching and localization. Our approach using the magnetic field achieves reasonable accuracy outdoors and saves 45-55% power over GPS. Our median error indoors is 0.3m using the magnetic field or CSI, which is up to 75% better than the point-based localization scheme even with the knowledge of the current segment.

Finally, we focus on building applications that leverage accurate location information in areas where location information has not been typically readily available, *e.g.*, within malls or grocery stores where GPS is not available. To analyze people's behavior in these areas can be immensely useful for various applications like a shopping helper, targeted ads, indoor navigation and friend discovery. To understand people's behavior, localizing them accurately within these spaces is critical. This is challenging because many of the assumptions that typical indoor localization schemes make, may not be valid here, *e.g.*, store floor maps may not be made available and moreover they may change dynamically making it hard to maintain them up-to-date. Useful information like Wi-Fi AP locations within the store, Wi-Fi and other finger-print data maybe unavailable too. While it is possible that some of this data maybe available within participating stores to build such a service, it is desirable to study the person's behavior within other stores too, to build a rich user profile. To tackle these challenges and provide an always on localization service to enable physical analytics, we present AutoLayout, that

simultaneously localizes users as well as builds product maps, based on Wi-Fi, inertial sensor and video data from smart glass users. Moreover, we present algorithms to automatically detect user behaviors such as: walking, dwelling, gazing and reaching-out. Through our evaluation in two large retail stores in the United States, we show the effectiveness of our physical analytics system — ThirdEye, despite not requiring any inputs from the users themselves.

6.2 Future Work

While this dissertation takes significant steps towards localization and enabling physical analytics in retail spaces, several interesting questions remain. First, we need to apply the methods for tracking physical browsing presented in this dissertation, on a large data-set of shoppers, to see patterns that are representative of a more diverse population. For example, looking at a large data-set of users could give us insights about better store layouts, better times and occasions for sales and discounts, or even sections of the store that would need more store representatives to help out shoppers.

Further, leveraging the techniques for localizing users, and understanding their interests, to build and deploy useful applications like in-store shopping guides, shopping list reminders to improve the shopping experience of the users is an interesting and practically useful area of work. Moreover, combining the physical browsing data with web analytics and leveraging the synergy is an important step that would help provide users with a holistic shopping experience. For example, if a user has researched online about cold or allergy related

problems, he could be alerted to pick up Kale, a known source of vitamin C as he walks through the vegetable section in a grocery store. Such an application would not only benefit the businesses but also the users to lead a more comfortable and healthy life.

Finally, beyond shopping, localization and the techniques for physical analytics presented in this dissertation are useful in many other contexts. For example, smart-phone/wearable device applications for localizing and understanding the activities of elderly people, may help them live independently while being monitored for any necessary emergency care. In an office environment, if mobile devices can combine information from a user's calendar and detect that he/she is in a meeting room for a meeting or a presentation, they could automatically go into the silent mode.

Building such applications is challenging as they need to be designed to work for a diverse set of users, usage patterns and environments on the resource constrained mobile devices. Nevertheless, pushing for these applications to come into wide-spread use is really the ultimate goal of the work presented in this dissertation.

Bibliography

[1] Bing Vision. http://en.wikipedia.org/wiki/Bing_Vision.

[2] Code for TextonBoost. http://jamie.shotton.org/work/code/TextonBoost.zip.

[3] Google search by image. http://www.google.com/insidesearch/features/images/searchbyimage.html.

[4] Nokia Point and Find. https://betalabs.nokia.com/trials/nokia-point-and-find.

[5] Tobii. http://www.tobii.com/.

[6] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *Proc. of ICCV*, 2009.

[7] Alert systems. http://www.alertsystems.org/.

[8] Moustafa Alzantot and Moustafa Youssef. Crowdinside: automatic construction of indoor floorplans. In *Proc. of the 20th International Conference on Advances in Geographic Information Systems*, 2012.

[9] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *Proc. of Mobicom*, 2009.

[10] Aline Baggio and Koen Langendoen. Monte carlo localization for mobile wireless sensor networks. *Ad Hoc Netw.*, 6(5), 2008.

[11] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proc. of IEEE INFOCOM*, 2000.

[12] Battery Level API. `http://developer.android.com/training/monitoring-device-state/battery-monitoring.html`.

[13] L. Benmohamed, P. Chimento, B. Doshi, B. Henrick, and I. J. Wang. Sensor network design for underwater surveillance. In *Proc. of Military Communications Conference (MILCOM)*, 2006.

[14] P. Biswas, T. C. Liang, K. C. Toh, and Y. Ye. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*.

[15] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proc. of 3rd IPSN*, 2004.

[16] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of ACM MobiCom*, Oct. 1998.

[17] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5), Oct. 2000.

[18] Cabspotting project. http://www.cabspotting.com.

[19] A. Campbell et al. The Rise of People-Centric Sensing. *IEEE Internet Computing*, Jul/Aug 2008.

[20] E. Candes and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies. *IEEE Trans. on Information Theory*, 2006.

[21] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: application driver for wireless communications technology. In *Proc. of ACM SIGCOMM Workshop on Data Communications*, Aug. 2001.

[22] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. Indoor Localization Without the Pain. In *Proc. of MobiCom*, 2010.

[23] T. Choudhury, S. Consolvo, B. Harrison, and J. Hightower. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, Apr-Jun 2008.

[24] Multidimensional scaling. http://en.wikipedia.org/wiki/Multidimensional_scaling.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[26] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, Sept. 2003.

[27] Andrew J. Davidson, Ian D. Reid, Nicholar D. Molton, and Olivier Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE PAMI*, 29(6), Jun 2007.

[28] Brian Ferris Dieter and Fox Neil Lawrence. Wifi-slam using gaussian process latent variable models. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[29] M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17, 2001.

[30] D. Donoho. For most large underdetermined systems of linear equations, the minimal L1-norm near-solution approximates the sparsest near-solution. `http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0approx.pdf`.

[31] D. Donoho. For most large underdetermined systems of linear equations, the minimal L1-norm solution is also the sparsest solution. `http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0EquivCorrected.pdf`.

[32] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 2006.

[33] Earth's Magnetic Field. `http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/magearth.html`.

[34] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G-S. Ahn, and A. T. Campbell. The BikeNet Mobile Sensing System for Cyclist Experience Mapping. In *Proc. of SenSys*, 2007.

[35] Euclid Analytics. Answers and insights for QSR, coffee, specialty retail, and large format stores. `http://euclidanalytics.com/`.

[36] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communication of the ACM*, 24(6):381–395, 1981.

[37] Google Glass. `http://www.google.com/glass/start/`.

[38] Global positioning system standard positioning service specification. *United States Coast Guard Navigation Center*, June 1995.

[39] S. Guha, R. N. Murty, and E. G. Sirer. Sextant: A unified framework for node and event localization in sensor networks. In *Proc. of ACM MobiHoc*, May 2005.

[40] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool release: gathering 802.11n traces with channel state information. *SIGCOMM Comput. Commun. Rev.*, 41(1), January 2011.

[41] Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. Energy-efficient surveillance system using wireless sensor networks. In *Proc. of ACM MobiSys*, 2004.

[42] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, Aug 2001.

[43] Berthold Horn and Brian Schunck. Determining Optical Flow. *Artifical Intelligence*, 17, 1981.

[44] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proc. of ACM MobiCom*, Sept. 2004.

[45] Human traces. `http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels`.

[46] iBeacon. `http://tech.fortune.cnn.com/2014/02/28/apples-ibeacon-signals-turning-point-for-mobile-engagement/`.

[47] Indoor Atlas. `http://web.indooratlas.com/web/WhitePaper.pdf`.

[48] Suman Jana and Sneha Kumar Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. In *Proc. of MobiCom*, 2008.

[49] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 30(5), 2002.

[50] Takayuki Kanda, Dylan F. Glas, Masahiro Shiomi, Hiroshi Ishiguro, and Norihiro Hagita. Who will be the customer?: A social robot that anticipates people's behavior from their trajectories. In *Proc. of UbiComp*, 2008.

[51] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *Proc. of SDM*, 2001.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proc. of NIPS*, 2012.

[53] S. Lee, C. Yoo, C. Min, and J. Song. Understanding customer malling behavior in an urban shopping mall using smartphones. In *Proc. of ACM Workshop on Mobile Systems for Computional Social Science (co-located with UbiComp)*, 2013.

[54] J.J Leonard and H. F Durrant-whyte. Simultanoues Map Building and Localization for an Autonomous Mobile Robot. In *IROS*, pages 1442–1447, 1991.

[55] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *Proc. of ECCV*, 2012.

[56] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(3), 1989.

[57] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 1982.

[58] David Lowe. Object Recognition from Local Scale-Invariant Features. In *Proc. of ICCV*, 1999.

[59] U. V. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[60] D. Madigan, E. Elnahrawy, and R. P. Martin. Bayesian indoor positioning systems. In *Proc. of IEEE INFOCOM*, Mar. 2005.

[61] R. Martens and L. Claesen. On-line signature verification by dynamic time-warping. In *Proc. of ICPR*, volume 3, Aug. 1996.

[62] M. H. T. Martins, H. Chen, and K. Sezaki. OTMCL: orientation tracking-based monte carlo localization for mobile sensor networks. In *Proc. of INSS*, 2009.

[63] Mesh connectivity layer. `http://research.microsoft.com/mesh/#software`.

[64] Localization for mobile sensor networks. http://www.cs.virginia.edu/mcl.

[65] MCS Index Table. http://mcsindex.com/.

[66] Marina Meila and Jianbo Shi. A random walk view of image segmentation. In *Proc. of AI and STATISTICS (AISTATS)*, 2001.

[67] minFunc. http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html.

[68] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proc. of ACM SenSys*, Nov. 2004.

[69] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[70] Nearbuy Systems. In-Store Mobile Commerce. http://www.nearbuysystems.com/.

[71] D. Niculescu and B. Nath. VOR base stations for indoor 802.11 positioning. In *Proc. of ACM MobiCom*, Sept. 2004.

[72] S. J. Orfanidis. *Introduction to Signal Processing*. Prentice Hall, 1996.

[73] Venkata N. Padmanabhan, Lili Qiu, and Helen Wang. Server-based inference of Internet performance. In *Proc. of IEEE INFOCOM*, Mar. 2003.

[74] Jayaguru Panda, Michael S. Brown, and C. V. Jawahar. Offline mobile instance retrieval with a small memory footprint. In *Proc. of ICCV*, 2013.

[75] Neal Patwari and Sneha K. Kasera. Robust location distinction using temporal link signatures. In *Proc. of ACM MobiCom*, 2007.

[76] Power monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[77] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proc. of ACM MobiCom*, Aug 2000.

[78] L. Rabiner, A. Rosenberg, and S. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(6), Dec. 1978.

[79] Anshul Rai, Krishna Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *Proc. of MobiCom*, 2012.

[80] Swati Rallapalli, Wei Dong, Lili Qiu, and Yin Zhang. Unified localization framework using trajectory signatures. In *Proc. of SIGMETRICS*, 2014.

[81] Swati Rallapalli, Aishwarya Ganesan, Krishna Chintalapudi, Venkata N. Padmanabhan, and Lili Qiu. Enabling Physical Analytics in Retail Stores Using Smart Glasses. In *Proc. of MobiCom*, 2014.

[82] Swati Rallapalli, Lili Qiu, Yin Zhang, and Yi chao Chen. Exploiting temporal stability and low-rank structure for localization in mobile networks. In *Proc. of MobiCom*, 2010.

[83] Patrick Robertson, Michael Angermann, and Bernhard Krach. Simultaneous Localization and Mapping for Pedestrians using only Foot-Mounted Inertial Sensors. In *Proc. of UbiComp*, 2009.

[84] Patrick Robertson, Maria Garcia Puyol, and Michael Angermann. Collaborative Pedestrian Mapping of Buildings: Using Inertial Sensors and FootSLAM. In *Proc. of ION GNSS*, Sep 2011.

[85] Masoomeh Rudafshani and Suprakash Datta. Localization in wireless sensor networks. In *Proc. of IPSN*, 2007.

[86] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1), Feb. 1978.

[87] A. Savvides, C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. of ACM MobiCom*, Jul. 2001.

[88] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proc. of WSNA'02*, Sep. 2002.

[89] Seattle bus traces. `http://crawdad.cs.dartmouth.edu/rice/ad_hoc_city`.

[90] Souvik Sen, Jeongkeun Lee, Kyu-Han Kim, and Paul Congdon. Avoiding multipath to revive inbuilding wifi localization. In *Proc. of ACM Mobisys*, 2013.

[91] Souvik Sen, Božidar Radunovic, Romit Roy Choudhury, and Tom Minka. You are facing the Mona Lisa: spot localization using phy layer information. In *Proc. of ACM MobiSys*, 2012.

[92] Y. Shang and W. Ruml. Improved MDS-based localization. In *Proc. of IEEE INFOCOM*, Apr. 2004. `http://www.ieee-infocom.org/2004/Papers/55_1.PDF`.

[93] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *Proc. of ACM MobiHoc*, Jun. 2003. `http://www.sigmobile.org/mobihoc/2003/papers/p201-shang.pdf`.

[94] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-markie: indoor pathway mapping made easy. In *Proc. of NSDI*, 2013.

[95] R. Smith, M. Self, and P. Cheeseman. Autonomous robot vehicles. chapter Estimating uncertain spatial relationships in robotics. Springer-Verlag, 1990.

[96] Laura Stampler. That Chocolate Chip Cookie in the Checkout Aisle Is Watching You. *Time NewsFeed*, October 2013. `http://newsfeed.time.com/2013/10/16/that-chocolate-chip-cookie-in-the-checkout-aisle-is-watching-you/`.

[97] D. C. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole. Research challenges in environmental observation and forecasting systems. In *Proc. of 6th International Conference on Mobile Computing and Networking*, 2000.

[98] E. Stevens-Navarro, V. Vivekanandan, and V. W. S. Wong. Dual and mixture monte carlo localization algorithms for mobile wireless sensor networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, 2007.

[99] Kalyan Pathapati Subbu, Brandon Gozick, and Ram Dantu. Indoor localization through dynamic time warping. In *SMC*, 2011.

[100] Underwater acoustic sensor networks. `http://www.ece.gatech.edu/research/labs/bwn/UWASN/`.

[101] Ilari Vallivaara, Janne Haverinen, Anssi Kemppainen, and J. Roning. Simultaneous localization and mapping using ambient magnetic field.

In *Proc. of Multisensor Fusion and Integration for Intelligent Systems*, 2010.

[102] VideoBlack. `http://glass-apps.org/videoblack-google-glass-app`.

[103] Volcano Monitoring, Harvard Sensor Networks Lab. `http://fiji.eecs.harvard.edu/Volcano`.

[104] F. Wang, L. Qiu, and S. Lam. Probabilistic region-based localization for wireless networks. *ACM Mobile Computing and Communications Review (MC2R) Special Issue on Localization*, Jan. 2007.

[105] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: unsupervised indoor localization. In *Proc. of ACM MobiSys*, 2012.

[106] Zizhou Wang, Song Zheng, Stephen Boyd, and Yinyu Ye. Further relaxations of the SDP approach to sensor network localization. `http://www.stanford.edu/~yyye/relaxationsdp9.pdf`.

[107] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Trans. Information Systems*, Jan. 1992.

[108] Wikipedia. Nearest-neighbor interpolation. `http://en.wikipedia.org/wiki/Nearest-neighbor_interpolation`.

[109] Wikipedia. Newton's method in optimization. http://en.wikipedia.org/wiki/Newton's_method_in_optimization.

[110] Wikipedia. Quasi-Newton method. http://en.wikipedia.org/wiki/Quasi-Newton_method.

[111] J. Xiong and K. Jamieson. ArrayTrack: A Fine-Grained Indoor Location System. In *Proc. of HotMobile*, 2012.

[112] Chenren Xu, Bernhard Firner, Yanyong Zhang, Richard Howard, and Jun Li. Exploiting human mobility trajectory information in indoor device-free passive tracking. In *Proc. of ACM IPSN*, 2012.

[113] Zheng Yang, Chenshu Wu, and Yunhao Liu. Locating in fingerprint space: wireless indoor localization with little human intervention. In *Proc. of ACM MobiCom*, 2012.

[114] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *Proc. of ACM MobiCom*, Sept. 2003.

[115] C. You, C. Wei, Y. Chen, H. Chu, and M. Chen. Using mobile phones to monitor shopping time at physical stores. *IEEE Pervasive Computing*, 2011.

[116] M Youssef and A Agrawala. The Horus WLAN Location Determination System. In *Proc. of MobiSys*, 2005.

[117] Moustafa Youssef and Ashok Agrawala. The Horus WLAN location determination system. In *Proc. of ACM MobiSys*, 2005.

[118] Zebranet traces. `http://crawdad.cs.dartmouth.edu/princeton/zebranet`.

[119] Junxing Zhang, Mohammad Hamed Firooz, Neal Patwari, and Sneha Kumar Kasera. Advancing wireless link signatures for location distinction. In *Proc. of ACM MobiCom*, 2008.

[120] Yin Zhang, Matthew Roughan, Walter Willinger, and Lili Qiu. Spatio-temporal compressive sensing and Internet traffic matrices. In *Proc. of ACM SIGCOMM*, Aug. 2009.

[121] Hongzi Zhu, Minglu Li, Yanmin Zhu, and Lionel M. Ni. Hero: Online real-time vehicle tracking. *IEEE Trans. Parallel Distrib. Syst.*, 20(5), 2009.